Magelis XBT GT, XBT GK HMI Controller

Programming Guide

04/2012

EIO0000000638.04

www.schneider-electric.com

Schneider Electric

The information provided in this documentation contains general descriptions and/or technical characteristics of the performance of the products contained herein. This documentation is not intended as a substitute for and is not to be used for determining suitability or reliability of these products for specific user applications. It is the duty of any such user or integrator to perform the appropriate and complete risk analysis, evaluation and testing of the products with respect to the relevant specific application or use thereof. Neither Schneider Electric nor any of its affiliates or subsidiaries shall be responsible or liable for misuse of the information contained herein. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without express written permission of Schneider Electric.

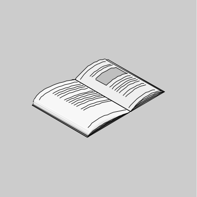
All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components.

When devices are used for applications with technical safety requirements, the relevant instructions must be followed.

Failure to use Schneider Electric software or approved software with our hardware products may result in injury, harm, or improper operating results.

Failure to observe this information can result in injury or equipment damage.

Table of Contents



Safety Information

Important Information

NOTICE

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

⚡ The addition of this symbol to a Danger safety label indicates that an electrical hazard exists, which will result in personal injury if the instructions are not followed.

⚠ This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

⚠ DANGER
DANGER indicates an imminently hazardous situation which, if not avoided, **will result in** death or serious injury.

⚠ WARNING
WARNING indicates a potentially hazardous situation which, if not avoided, **can result in** death or serious injury.

⚠ CAUTION

CAUTION indicates a potentially hazardous situation which, if not avoided, **can result in** minor or moderate injury.

NOTICE

NOTICE is used to address practices not related to physical injury.

PLEASE NOTE

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and its installation, and has received safety training to recognize and avoid the hazards involved.

About the Book

At a Glance

Document Scope

The purpose of this document is to:

- show you how to program and operate your XBT GT/GK HMI Controller,
- help you to understand how to program your XBT GT/GK HMI Controller functions,
- help you to become familiar with the XBT GT/GK HMI Controller functions.

Read and understand this document and all related documents before installing, operating or maintaining your XBT GT/GK HMI Controller

Validity Note

This document has been updated with the release of SoMachine V3.1.

Related Documents

Title of Documentation	Reference Number

SoMachine Programming Guide	EIO0000000067 (ENG);
	EIO0000000069 (FRE);
	EIO0000000068 (GER);
	EIO0000000071 (SPA);
	EIO0000000070 (ITA);
	EIO0000000072 (CHS)

Magelis XBTGT, XBTGK, XBTGH Hardware Guide	35010372 (ENG);
	35010373 (FRE);
	35010374 (GER);
	35010375 (SPA);
	35010798 (ITA);
	35010376 (CHS)

Magelis XBT Gx HMI Controller System Functions and Variables XBT PLCSystem Library Guide	EIO0000000626 (ENG);
	EIO0000000627 (FRE);
	EIO0000000628 (GER);
	EIO0000000629 (SPA);
	EIO0000000630 (ITA);
	EIO0000000631 (CHS)

SoMachine Modbus and ASCII Read/Write Functions PLCCommunication Library Guide	EIO0000000361 (ENG);
	EIO0000000742 (FRE);
	EIO0000000743 (GER);
	EIO0000000744 (SPA);
	EIO0000000745 (ITA);
	EIO0000000746 (CHS)

You can download these technical publications and other technical information from our website at www.schneider-electric.com.

Product Related Information

⚠ WARNING

LOSS OF CONTROL

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.[1]
- Each implementation of this equipment must be individually and thoroughly tested for proper operation before being placed into service.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

[1] For additional information, refer to NEMA ICS 1.1 (latest edition), "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control" and to NEMA ICS 7.1 (latest edition), "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Only use software approved by Schneider Electric for use with this equipment.
- Update your application program every time you change the physical hardware configuration.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

User Comments

We welcome your comments about this document. You can reach us by e-mail at techcomm@schneider-electric.com.

Starting with a New Project

1

Introduction

This chapter describes how to create a project with the XBT GT/GK HMI Controller and how to add devices.

What’s in this Chapter?

This chapter contains the following sections:

1.1 New Project

Introduction

This section will guide you through creating a new XBT GT/GK HMI Controller project.

What's in this Section?

This section contains the following topics:

Creating a new Project

Introduction

This section describes the general characteristics of the XBT GT/GK HMI Controller and how to create a new SoMachine project. Refer to *Manage your project (see SoMachine, Programming Guide)* for additional information on the project management.

XBT GT/GK HMI Controller Main Characteristics

The following table lists the main characteristics for the XBT GT/GK HMI Controller:

Controller	Display Type	Ethernet Interface	Serial Interface	USB Interface	CF Card Interface
XBT GT 1105	QVGA/STN Amber	No	Yes (1)	Yes	No
XBT GT 1135	QVGA/STN Amber	Yes	Yes (1)	Yes	No
XBT GT 1335	QVGA/STN Amber	Yes	Yes (1)	Yes	No
XBT GT 2110	QVGA/STN Monochrome	No	No	Yes	No
XBT GT 2120	QVGA/STN Monochrome	No	Yes (1)	Yes	Yes
XBT GT 2130	QVGA/STN Monochrome	Yes	Yes (1)	Yes	Yes
XBT GT 2220	QVGA/STN Color	No	No	Yes	Yes
XBT GT 2330	QVGA/TFT Color	Yes	Yes (1)	Yes	Yes
XBT GT 4230	VGA/STN Color	Yes	Yes (1)	Yes	Yes
XBT GT 4330	VGA/TFT Color	Yes	No	Yes	Yes
XBT GT 4340	VGA/TFT Color	Yes	Yes (1)	Yes	Yes
XBT GT 5230	VGA/STN Color	Yes	Yes (1)	Yes	Yes
XBT GT 5330	VGA/TFT Color	Yes	No	Yes	Yes
XBT GT 5340	VGA/TFT Color	Yes	Yes (1)	Yes	Yes
XBT GT 6330	SVGA/TFT Color	Yes	Yes (1)	Yes	Yes
XBT GT 6340	SVGA/TFT Color	Yes	Yes (1)	Yes	Yes
XBT GT 7340	XGA/TFT Color	Yes	Yes (1)	Yes	Yes
XBT GK 2120	QVGA/STN Monochrome	No	Yes (1)	Yes	Yes

Controller	Display Type	Ethernet Interface	Serial Interface	USB Interface	CF Card Interface
XBT GK 2330	QVGA/TFT Color	Yes	Yes (1)	Yes	Yes
XBT GK 5330	VGA/TFT Color	Yes	Yes (1)	Yes	Yes
Legend					
1	RS232/RS422/RS485 serial interface. SUB-D 9-pin connector				

NOTE: Refer to the *Controller specifications (see Magelis XBT GT, XBT GK, XBT GH, Hardware Guide)* for addtional information.

Creating a New Project

To create a new project, you must add a controller to the project **Devices** window. Refer to Devices Tree Description *(see page 15)* to see the hardware structure of your project, and refer to Adding an XBT GT/GK HMI Controller *(see page 18)* to add a controller to your project.

Active Application

The active application is displayed in bold print in the **Devices** window. When working on a project that contains several applications, check that the application you are currently working on is activated. Certain commands (for example, the **Build** command) are by default executed on the active application.

To activate an application, right-click its entry in the **Devices** window and select **Set Active Application** from the context menu.

NOTE: Using **Set Active Application** during multiple application controls, (not HMI applications) changes the description of several commands in the **Build** menu, in order to refer to the new active application.

Devices Tree Description

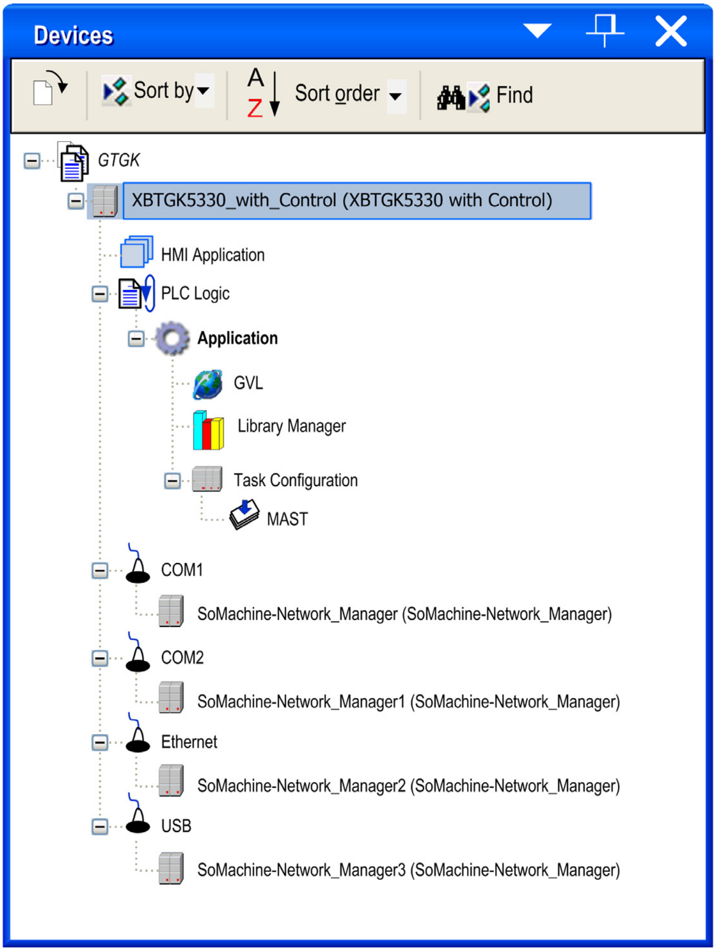
Introduction

The Devices tree shows the hardware objects such as the controller, field bus nodes, and I/O module, and also shows resources that are needed to run your application, such as tasks, POUs, and global variable lists.

Refer to CoDeSys Online help for more information on the Devices Tree.

View of the Devices Window

The **Devices** window contains the device tree and describes the hardware configuration of your project, as shown in the next picture:



Devices

- GTGK
  - XBTGK5330_with_Control (XBTGK5330 with Control)
    - HMI Application
    - PLC Logic
      - Application
        - GVL
        - Library Manager
        - Task Configuration
          - MAST
    - COM1
      - SoMachine-Network_Manager (SoMachine-Network_Manager)
    - COM2
      - SoMachine-Network_Manager1 (SoMachine-Network_Manager)
    - Ethernet
      - SoMachine-Network_Manager2 (SoMachine-Network_Manager)
    - USB
      - SoMachine-Network_Manager3 (SoMachine-Network_Manager)

Devices Tree Description

The following table describes the items in the **Devices** tree:

Item	Description
HMI Application	Used to configure the HMI part of your project
PLC Logic	Shows elements of your application: ● **GVL:** Global Variables List ● **Library Manager:** Application Library Manager ● **Task Configuration:** MAST and other task configuration information
COM1 / COM2	Embedded communication functions for Serial Line *(see page 81)* communication.
Ethernet	Embedded communication functions for Ethernet *(see page 71)* communication.
USB	Embedded communication functions for USB communication.

1.2 Adding Devices to the Project

Introduction

This section shows you how to add devices to your project.

What’s in this Section?

This section contains the following topics:

Adding an XBT GT/GK HMI Controller

Introduction

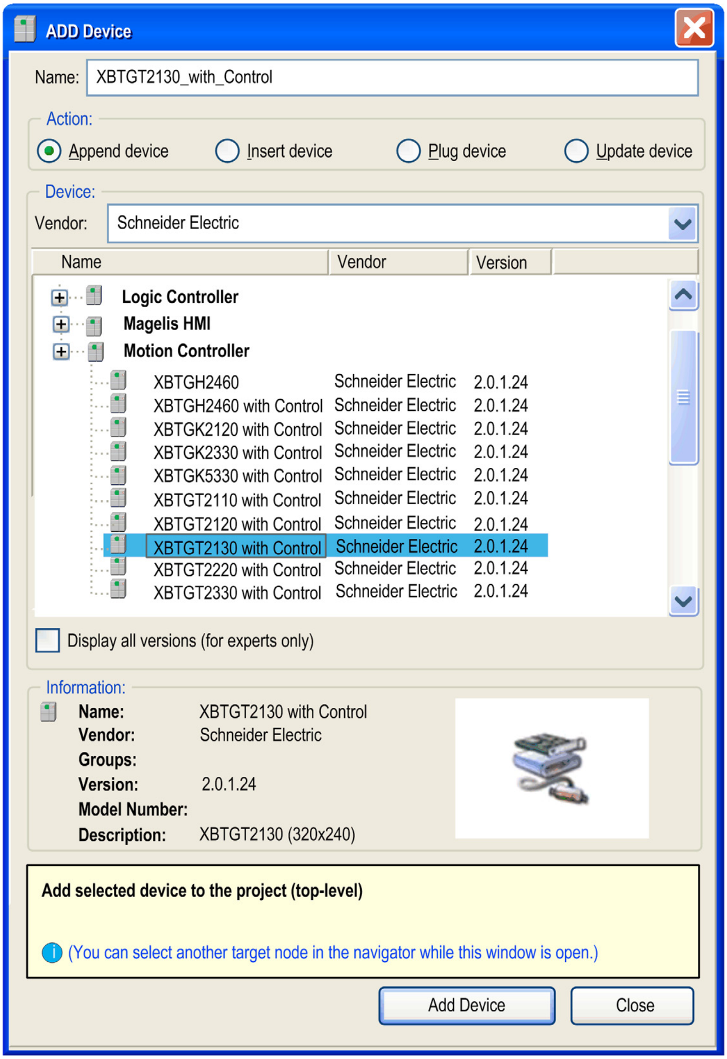
The following paragraphs explain how to add the XBT GT/GK HMI Controller to a SoMachine project.

Adding the XBT GT/GK HMI Controller to the Project Tree

Add the XBT GT/GK HMI Controller (available in the **Devices** window) to the project tree, using the default settings that are specific to each device. To set the parameters of the controller to your individual requirements, configure the devices via the **Devices** window.

The following table shows you how to add the XBT GT/GK HMI Controller to your project configuration using the **Add Device** window:

Step	Action
1	Right-click the project node in the **Devices** window and select **Add Device**. **Tip:** Alternatively, click **Project** → **Add Devices** from the main menu bar.

Step	Action
2	In the **Add Devices** window, select the XBT GT device, as shown in the figure below.  ADD Device Name: XBTGT2130_with_Control Action: ● Append device ○ Insert device ○ Plug device ○ Update device Device: Vendor: Schneider Electric Name / Vendor / Version ⊞ Logic Controller ⊞ Magelis HMI ⊞ Motion Controller XBTGH2460 / Schneider Electric / 2.0.1.24 XBTGH2460 with Control / Schneider Electric / 2.0.1.24 XBTGK2120 with Control / Schneider Electric / 2.0.1.24 XBTGK2330 with Control / Schneider Electric / 2.0.1.24 XBTGK5330 with Control / Schneider Electric / 2.0.1.24 XBTGT2110 with Control / Schneider Electric / 2.0.1.24 XBTGT2120 with Control / Schneider Electric / 2.0.1.24 XBTGT2130 with Control / Schneider Electric / 2.0.1.24 XBTGT2220 with Control / Schneider Electric / 2.0.1.24 XBTGT2330 with Control / Schneider Electric / 2.0.1.24 □ Display all versions (for experts only) Information: **Name:** XBTGT2130 with Control **Vendor:** Schneider Electric **Groups:** **Version:** 2.0.1.24 **Model Number:** **Description:** XBTGT2130 (320x240) **Add selected device to the project (top-level)** ⓘ (You can select another target node in the navigator while this window is open.) Add Device Close **Note**: To sort the devices by type in the **Add Device** window, select **Schneider Electric** in the **Vendor** list box.
3	Select the controller you want to add to your configuration.

Step	Action
4	Rename your device by typing a name in the **Name** field. **Note:** ● Do not use spaces or special characters (%, #). ● The length of the name cannot exceed 32 characters.
5	Click **Add Device** to add the device to your project. **Result:** The **Add Device** window re-opens.
6	To add another controller, repeat step 3 above. Otherwise, close the **Add Device** window.

NOTE: Another method to add a controller to your project is to use the *Graphical Configuration Editor (see SoMachine, Programming Guide)*, refer to *Adding and Deleting Devices (see SoMachine, Programming Guide)* for additional information.

Adding a CANopen Expansion Module

Introduction

You can add one of the following CANopen expansion module with the XBT GT/GK HMI Controller:

- XBT ZGCANM for standard application
- XBT ZGCANMS0 for Solution Architecture application

The CANbus node is automatically created. You can then add and configure further CANopen devices to the manager.

Adding a CANopen expansion is explained in CANopen Interface Configuration *(see page 74)*.

Libraries

2

Libraries

Introduction

The libraries of the controller provide functions such as function blocks, data types and global variables that can be used to develop your project. The default extension for a library is “.library”.

The **Library Manager** of SoMachine provides information about the libraries included in your project. You can also use the **Library Manager** to install new libraries.

Refer to the CoDeSys Online-Help for additional information about the **Library Manager**.

XBT GT/GK HMI Controller Libraries

When you select an XBT GT/GK HMI Controller for your application, SoMachine automatically loads the following libraries:

- **IoStandard:CmpIoMgr** configures types, access, parameters and help functions
- **Standard:** Bistable function blocks, counter, miscellaneous, string functions, timer and trigger
- **Util:** Analog monitors, BCD Conversions, Bit/Byte functions, controller datatypes, function manipulators, mathematical functions and signals
- **PLCCommunication:** Enables communication and it is common to all controller
- **XBT PLCSystem:** Refer to *XBT PLCSystem Library*

NOTE: The XBT GT/GK HMI Controller Libraries can be accessed from the **Devices** window only if you choose a XBT GT/GK HMI Controller with control.

Supported Standard Data Types

3

Introduction

This chapter provides the supported variables and explains how to exchange data between SoMachine (controller part) and Vijeo-Designer (HMI part).

What's in this Chapter?

This chapter contains the following topics:

Supported Variables

Supported Variables Types

The following table provides the XBT GT/GK HMI Controller supported variables types:

Controller Data type	Lower limit	Upper limit	Information content	Bidirectional Variable (SoMachine/Vijeo-Designer)
BOOL	False	True	1 Bit	Yes
DINT	-2,147,483,648	2,147,483,647	32 Bit	Yes
INT	-32,768	32,767	16 Bit	Yes
UINT	0	65,535	16 Bit	Yes
WORD	0 (hex)	FFFF (hex)	16 Bit	Yes
UDINT	0	4,294,967,295	32 Bit	Yes
DWORD	0 (hex)	FFFFFFFF (hex)	32 Bit	Yes
SINT	-128	127	8 Bit	Yes
USINT	0	255	8 Bit	Yes
BYTE	00 (hex)	FF (Hex)	8 Bit	Yes
REAL	-3.402824^{38}	3.402824^{38}	32 Bit	Yes
WSTRING	1 character	255 characters	1 character = 1 word	Yes
STRING	1 character	255 characters	1 character = 1 byte	Yes

Refer to:

- *Single Variable Definition (see SoMachine, Programming Guide)* for additional information on SoMachine/HMI data exchange
- CoDeSys Online-Help

Using Array and Structure Elements for Data Exchange

You can use array and structure elements for data exchange between the controller side (SoMachine) and the HMI side (Vijeo-Designer). However, you cannot exchange whole arrays and structures at once.

For example:

- If `A` is an array, you can exchange an element of the array (`A[0],A[1],...,A[i]`) but not the entire array.
- The same rule applies to structure element, you can exchange an element of the structure (`StructureName.ElementName`) but not the entire structure.

Variables Exchange

Introduction

You can exchange variables with the XBT GT/GK HMI Controller range between SoMachine and Vijeo-Designer by publishing them.

Controller and HMI Data Exchange

For variable exchange between the controller and HMI parts, perform the following steps:

- Create variables in the controller part.
- Publish the variables by defining them as **Symbols** in the controller part. They are now available in the HMI part as SoMachine variables.

Refer to SoMachine Single Variable Definition *(see SoMachine, Programming Guide)* for additional information on how to publish variables.

Once symbols have been transferred toVijeo-Designer (the HMI part of your application), it is usually not necessary to make the transfer every time you call Vijeo-Designer. If you later add or modify symbols in your SoMachine application after having initially transfered the symbols, you must again transfer symbols to Vijeo-Designer.

⚠ **WARNING**

UNINTENDED EQUIPMENT OPERATION

After adding or modifying symbols shared between the XBT GT/GK HMI Controller and other controllers, you must:

- Update the Vijeo-Designer application,
- Download the updated application into the XBT GT/GK HMI Controller.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Refer to *SoMachine-HMI Data Exchange (see SoMachine, Programming Guide)* for additional information on how to exchange variables.

Controller Memory Mapping

4

Introduction

This chapter provides the maximum size of an application for a XBT GT/GK HMI Controller, the size of the RAM , the located variables area and the libraries.

What's in this Chapter?

This chapter contains the following topics:

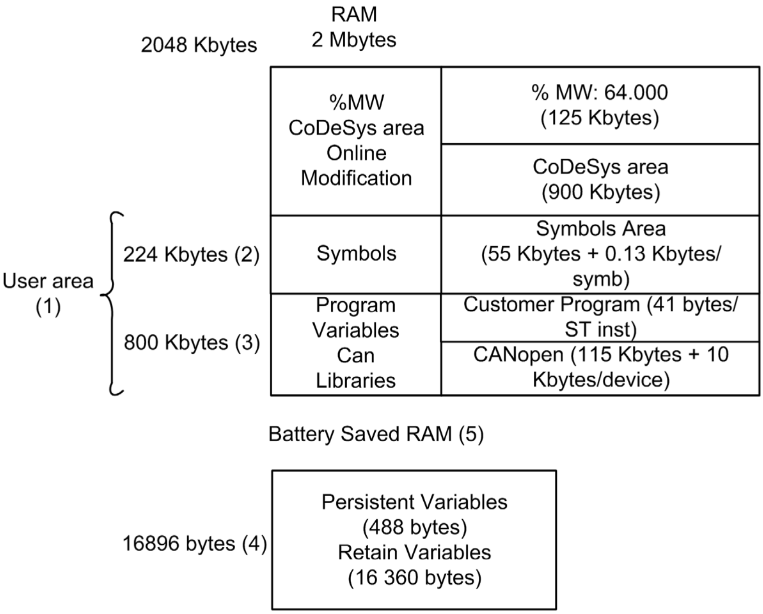
Memory Mapping

Introduction

This section provides the RAM (Random Access Memory) size for each area of the XBT GT/GK HMI Controller.

XBT GT/GK HMI Controller Memory

This figure shows different types of areas and their corresponding size for the XBT GT/GK HMI Controller memory:



Legend:	
(1)	1024 Kbytes must be shared among application, symbol area, CANopen, without any limit inside these 1024 Kbytes.
(2)	The symbols area size is not checked at build time.
(3)	800 Kbytes is checked at build time.
(4)	Not all of the 16 896 bytes are available for the customer application because some libraries may use Retain variables.
(5)	The battery is rechargeable and cannot be replaced.

Memory of the Supported Applications

The following table lists the memory capacity of supported applications:

Types of application or variables:	**Memory Capacity:**
User Application	1 Mbyte
Retain variables[1]	16 360 bytes (total of 16384 bytes where 24 bytes are reserved)
Persistent variables[1]	488 bytes (total of 512 bytes where 24 bytes are reserved)
Legend:	
(1)	Refer to the CoDeSys Online-Help for additional information on variables

Controllers and HMI Address Mapping Differences

Introduction

This following paragraphs provide instructions for double words and bits addressing between controller and the XBT GT/GK HMI Controller.

If you do not program your application to recognize the differences in address mapping between the controller and HMI parts, the controller and the HMI will not communicate correctly and it will be possible for incorrect values to be written to memory areas responsible for output operations.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Program your application to translate between the memory mapping used by the controller part and that used by the HMI part.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Memory Data Exchange

When the controller and the XBT GT/GK HMI Controller are connected, the data exchange uses simple word requests.

There is an overlap on simple words of the XBT GT/GK HMI Controller memory while using double words but not for the controller memory:

<table>
<tr><th colspan="4">Controller addressing</th><th></th><th colspan="4">HMI addressing</th></tr>
<tr><td>%MX0.7...%MX0.0</td><td>%MB0</td><td rowspan="2">%MW0</td><td rowspan="4">%MD0</td><td rowspan="6">The double word is split into two simple words.</td><td rowspan="2"></td><td rowspan="4">%MD0</td><td rowspan="2">%MW0</td><td>%MW0:X7...%MW0:X0</td></tr>
<tr><td>%MX1.7...%MX1.0</td><td>%MB1</td><td>%MW0:X15...%MW0:X8</td></tr>
<tr><td>%MX2.7...%MX2.0</td><td>%MB2</td><td rowspan="2">%MW1</td><td rowspan="4">%MD1</td><td rowspan="2">%MW1</td><td>%MW1:X7...%MW1:X0</td></tr>
<tr><td>%MX3.7...%MX3.0</td><td>%MB3</td><td>%MW1:X15...%MW1:X8</td></tr>
<tr><td>%MX4.7...%MX4.0</td><td>%MB4</td><td rowspan="2">%MW2</td><td rowspan="4">%MD1</td><td rowspan="4">%MD2</td><td rowspan="2">%MW2</td><td>%MW2:X7...%MW2:X0</td></tr>
<tr><td>%MX5.7...%MX5.0</td><td>%MB5</td><td>%MW2:X15...%MW2:X8</td></tr>
<tr><td>%MX6.7...%MX6.0</td><td>%MB6</td><td rowspan="2">%MW3</td><td colspan="2" rowspan="2">----------------------></td><td rowspan="2">%MW3</td><td>%MW3:X7...%MW3:X0</td></tr>
<tr><td>%MX7.7...%MX7.0</td><td>%MB6</td><td>%MW3:X15...%MW3:X8</td></tr>
</table>

In order to have a match between the XBT GT/GK HMI Controller memory area and the controller memory area, the ratio between double words of XBT GT/GK HMI Controller memory and the double words of controller memory is 2.

Examples

The following gives examples of memory match for the double words:

- %MD2 memory area of the XBT GT/GK HMI Controller corresponds to %MD1 memory area of the controller.
- %MD20 memory area of the XBT GT/GK HMI Controller corresponds to %MD10 memory area of the controller.

The following gives examples of memory match for the bits:

- %MW0:X9 memory area of the XBT GT/GK HMI Controller corresponds to %M1.1 memory area of the controller because the simple words are split in 2 distinct bytes in the controller memory.

Tasks

5

Introduction

The Task Configuration node in the SoMachine device tree allows you to define one or several tasks to control the execution of your application program.

The task types available are:

- Cyclic
- Freewheeling
- Event

This chapter begins with an explanation of these task types and provides information regarding the maximum number of tasks, the default task configuration, and task prioritization. In addition, this chapter introduces the system and task watchdog functions and explains their relationship to task execution.

What's in this Chapter?

This chapter contains the following topics:

Maximum Number of Tasks

Maximum Number of Tasks

The maximum number of tasks you can define for the XBT GT/GK HMI Controller are:

- Total number of tasks = 3
- Cyclic tasks = 3
- Freewheeling tasks = 1
- Event tasks = 2

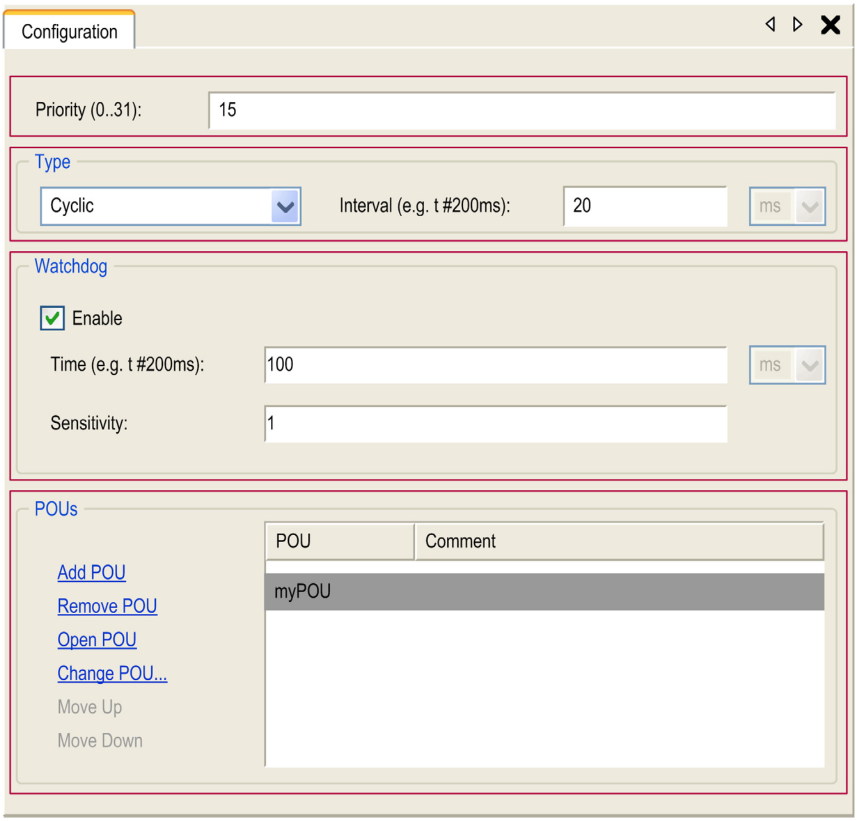
Task Configuration Screen

Screen Description

The following screen allows you to configure the tasks. Double click on the task that you want to configure in the device tree of the **Devices** window to access this screen.

Each configuration task has its own parameters which are independent of the other tasks.

The **task configuration** window is composed of 4 parts:


Configuration

Priority (0..31): 15

Type

Cyclic Interval (e.g. t #200ms): 20 ms

Watchdog

☑ Enable

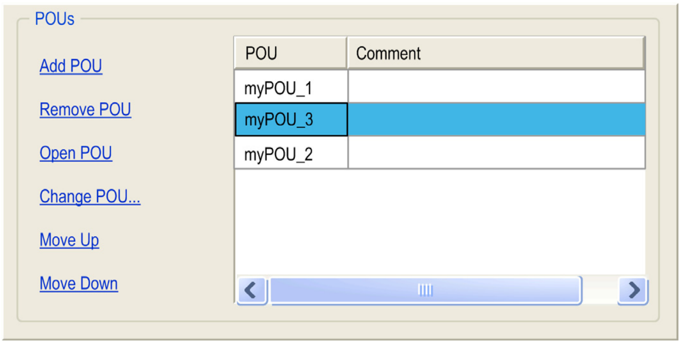
Time (e.g. t #200ms): 100 ms

Sensitivity: 1

POUs

POU	Comment
myPOU	

The following table describes the fields of the **Task Configuration** screen:

Field Name	Definition
Priority	You can configure the priority of each task with a number between 0 and 31 (0 is the highest priority, 31 is the lowest). Only one task at a time can be running. The priority determines when the task will run: ● a higher priority task will preempt a lower priority task ● tasks with same priority will run in turn (2 ms time-slice) **NOTE:** Do not assign tasks with the same priority. If there are yet other tasks that attempt to preempt tasks with the same priority, the result could be indeterminate and unpredictable. For more information, Task Priorities *(see page 43)*.
Type	4 types of task are available: ● Cyclic *(see page 40)* ● Freewheeling *(see page 41)* ● Event *(see page 41)*
Watchdog *(see page 42)*	To configure the watchdog, you must define two parameters: ● Time: enter the timeout before watchdog execution. ● Sensitivity: defines the number of expirations of the watchdog timer before the Controller stops in Exception mode.
POUs *(see SoMachine, Programming Guide)*	The list of **POU**s (Programming Organization Units) controlled by the task is defined in the task configuration window  POUs Add POU Remove POU Open POU Change POU... Move Up Move Down POU | Comment myPOU_1 myPOU_3 myPOU_2 ● To add a POU linked to the task, use the command **Add Pou** and select the POU in the **Input Assistant** editor. ● To remove a POU from the list, use the command **Remove POU**. ● The command **Open POU** opens the currently selected POU editor. ● To replace the currently selected POU of the list by another one, use the command **Change POU...** ● **POU**s are executed in the order shown in the list. To move the **POU**s in the list, select a **POU** and use the command **Move Up** or **Move Down**. **NOTE:** You can create as many POUs as you want. An application with several small POUs, as opposed to one large POU, can improve the refresh time of the variables in online mode.

XBT GT/GK HMI Controller Cycle Time Management

The XBT GT/GK HMI Controller cycle time management is set with the following configuration:

- 50% for the control
- 50% for the HMI application

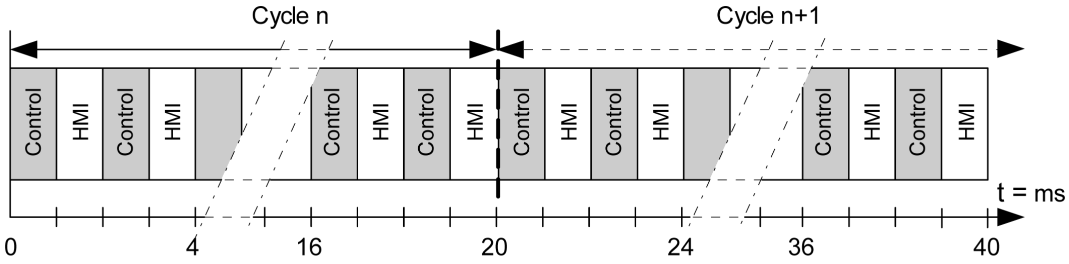
You should use a cycle time superior or equal to 20 ms.The period for the entire cycle must be a multiple of 4 ms (20, 24, 28, 32, 36 ms, etc.).

NOTE:

For XBTGC1100, 2120, and 2230 Embedded I/Os:

- There can be up to 4 ms latency between when an input gets a signal and when the controller gets this data.
- There can be up to 4 ms latency between when a variable is set and when the physical output actually changes state or value.

The following diagram shows an example of cycle time management between the control and HMI parts. In this example, the cycle time is set to 20 ms:



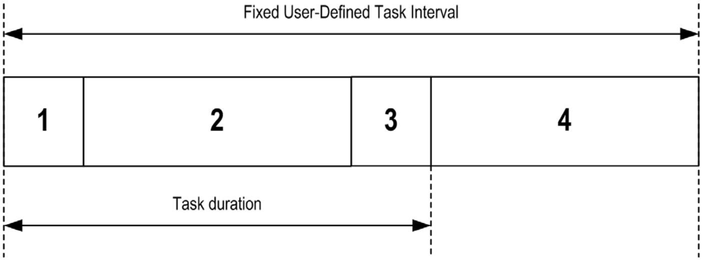
Task Types

Introduction

The following section describes the various task types available for your program, along with a description of the task type characteristics.

Cyclic Task

A Cyclic task is assigned at a fixed cycle time using the Interval setting in the Type section of Configuration sub-tab for that task. Each Cyclic task type executes as follows:



1. **Read Inputs:** The input states are written to the %I input memory variable and other system operations are executed.
2. **Task Processing:** The user code (POU, etc.) defined in the task is processed. The %Q output memory variable is updated according to your application program instructions but not written to the physical outputs during this operation.
3. **Write Outputs:** The %Q output memory variable is modified with any output forcing that has been defined, however, the writing of the physical outputs depends upon the type of output and instructions used. For more information on defining the Bus cycle task refer to CoDeSys online help. For more information on I/O behavior, refer to Controller States Detailed Description *(see page 53)*.
4. **Remaining Interval time:** The controller OS carries out system processing and any other lower priority tasks.

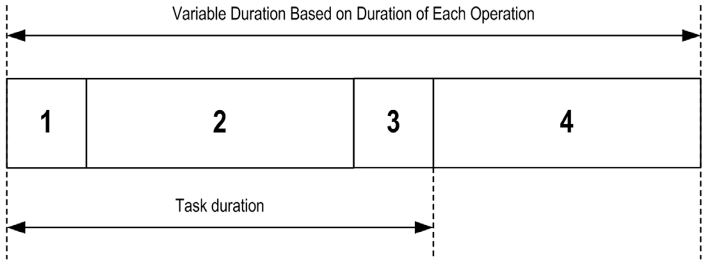
NOTE: If you define an insufficient period for a cyclic task, it will repeat immediately after the write of the outputs without executing other lower priority tasks or any system processing. This will affect the execution of all tasks and cause the controller to exceed the task watchdog limits (if set by a user), generating a task watchdog exception.

For the XBT GT/GK HMI Controller, system watchdog limits are not enforced.

NOTE: You can get and set the interval of a Cyclic Task by application using the **GetCurrentTaskCycle** and **SetCurrentTaskCycle** function.

Freewheeling Task

A Freewheeling task does not have a fixed duration. Each Freewheeling task type executes as follows:



1. **Read Inputs:** The input states are written to the %I input memory variable and other system operations are executed.
2. **Task Processing:** The user code (POU, etc.) defined in the task is processed. The %Q output memory variable is updated according to your application program instructions but not written to the physical outputs during this operation.
3. **Write Outputs:** The %Q output memory variable is modified with any output forcing that has been defined, however, the writing of the physical outputs depends upon the type of output and instructions used. For more information on defining the Bus cycle task refer to CoDeSys online help. For more information on I/O behavior, refer to Controller States Detailed Description *(see page 53)*.
4. **System Processing:** The controller OS carries out system processing and any other lower priority tasks. The length of the system processing period is set to 30 % of the total duration of the 3 previous operations (4 = 30 % x (1 + 2 + 3)). In any case, the system processing period won’t be lower than 3 ms.

Event Task

This type of task is event-driven and is initiated by a program variable. It starts at the rising edge of the boolean variable associated to the trigger event unless preempted by a higher priority task. In that case, the Event task will start as dictated by the task priority assignments.

For example, if you have defined a variable called `my_Var` and would like to assign it to an Event, select the **Event type** on the **Configuration** sub-tab and click on the **Input Assistant** button ... to the right of the **Event name** field. This will cause the **Input Assistant dialog** box to appear. In the **Input Assistant dialog** box, you navigate the tree to find and assign the `my_Var` variable.

System and Task Watchdogs

Introduction

Two types of watchdog functionality are implemented for the XBT GT/GK HMI Controller. These are:

- **System Watchdogs**: These watchdogs are defined in and managed by the controller OS (firmware). These are not configurable by the user.
- **Task Watchdogs**: Optional watchdogs that can be defined for each task. These are managed by your application program and are configurable in SoMachine.

System Watchdogs

The two system watchdogs common to the M2•8 and LMC series controllers for task overflow detection are not supported by the XBT GT/GK HMI Controller.

NOTE: If an application is sensitive to task time overrun, you should manually set the Task Watchdog.

Task Watchdogs

SoMachine allows you to configure an optional task watchdog for every task defined in your application program. (Task watchdogs are sometimes also referred to as software watchdogs or control timers in the SoMachine online help). When one of your defined task watchdogs reaches its threshold condition, an application error is detected and the controller enters the HALT state.

When defining a task watchdog, the following options are available:

- **Time**: This defines the allowable maximum execution time for a task. When a task takes longer than this the controller will report a task watchdog exception.
- **Sensitivity**: The sensitivity field defines the number of task watchdog exceptions that must occur before the controller detects an application error.

A task watchdog is configured on the Configuration sub-tab of the Task Configuration tab for the individual task. To access this tab, double-click on the task in the device tree.

NOTE: For more details on watchdogs, refer to the CoDeSys online help.

Task Priorities

Introduction

You can configure the priority of each task between 0 and 31 (0 is the highest priority, 31 is the lowest). Each task must have a unique priority.

⚠ **WARNING**

UNINTENDED EQUIPMENT OPERATION

Do not assign the same priority to different tasks.

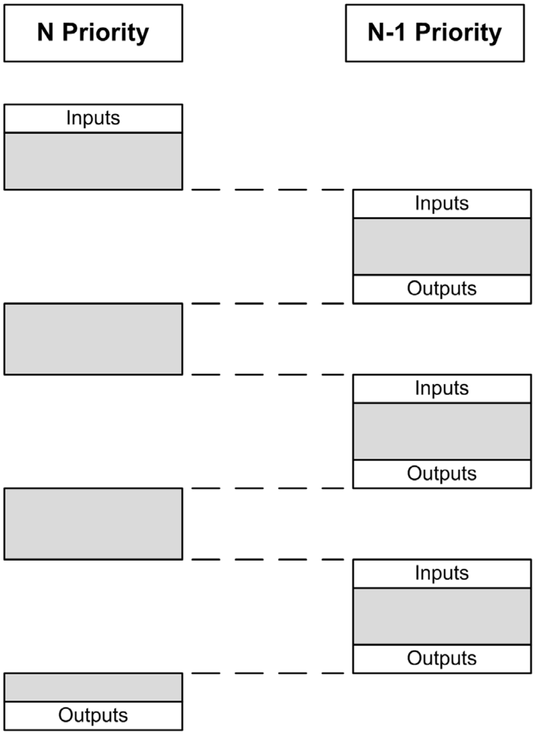
Failure to follow these instructions can result in death, serious injury, or equipment damage.

Task Priority Recommendations

- Priority 0 to 24: Controller tasks. Assign these priorities to tasks with a high real-time requirement.
- Priority 25 to 31: Background tasks. Assign these priorities to tasks with a low real-time requirement.

Task Preemption Due to Task Priorities

When a task cycle starts, it can interrupt any task with lower priority (task preemption). The interrupted task will resume when the higher priority task cycle is finished.



NOTE: If the same input is used in different tasks the input image may change during the task cycle of the lower priority task.

To improve the likelihood of proper output behavior during multitasking, an error is detected if outputs in the same byte are used in different tasks.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Map your inputs so that tasks do not alter the input images in an unexpected manner.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Default Task Configuration

Default Task Configuration

For the XBT GT/GK HMI Controller:

- The MAST task can be configured in Freewheeling or Cyclic mode. The MAST task is automatically created by default in Cyclic mode. Its preset priority is medium (15), its preset interval is 20 ms, and its task watchdog service is activated with a time of 100 ms and a sensitivity of 1. Refer to Task Priorities *(see page 43)* for more information on priority settings. Refer to System and Task Watchdogs *(see page 42)* for more information on watchdogs.

Designing an efficient application program is important in systems approaching the maximum number of tasks. In such an application, it can be difficult to keep the resource utilization below the system watchdog threshold. If priority reassignments alone are not sufficient to remain below the threshold, some lower priority tasks can be made to use fewer system resources if the SysTaskWaitSleep function is added to those tasks. For more information about this function, see the optional SysTask library of the system / SysLibs category of libraries.

NOTE: Do not delete or change the Name of the MAST task. If you do so, SoMachine detects an error when you attempt to build the application, and you will not be able to download it to the controller.

Controller States and Behaviors

6

Introduction

This chapter provides you with information on controller states, state transitions, and behaviors in response to system events. It begins with a detailed controller state diagram and a description of each state. It then defines the relationship of output states to controller states before explaining the commands and events that result in state transitions. It concludes with information about Remanent variables and the effect of SoMachine task programming options on the behavior of your system.

What's in this Chapter?

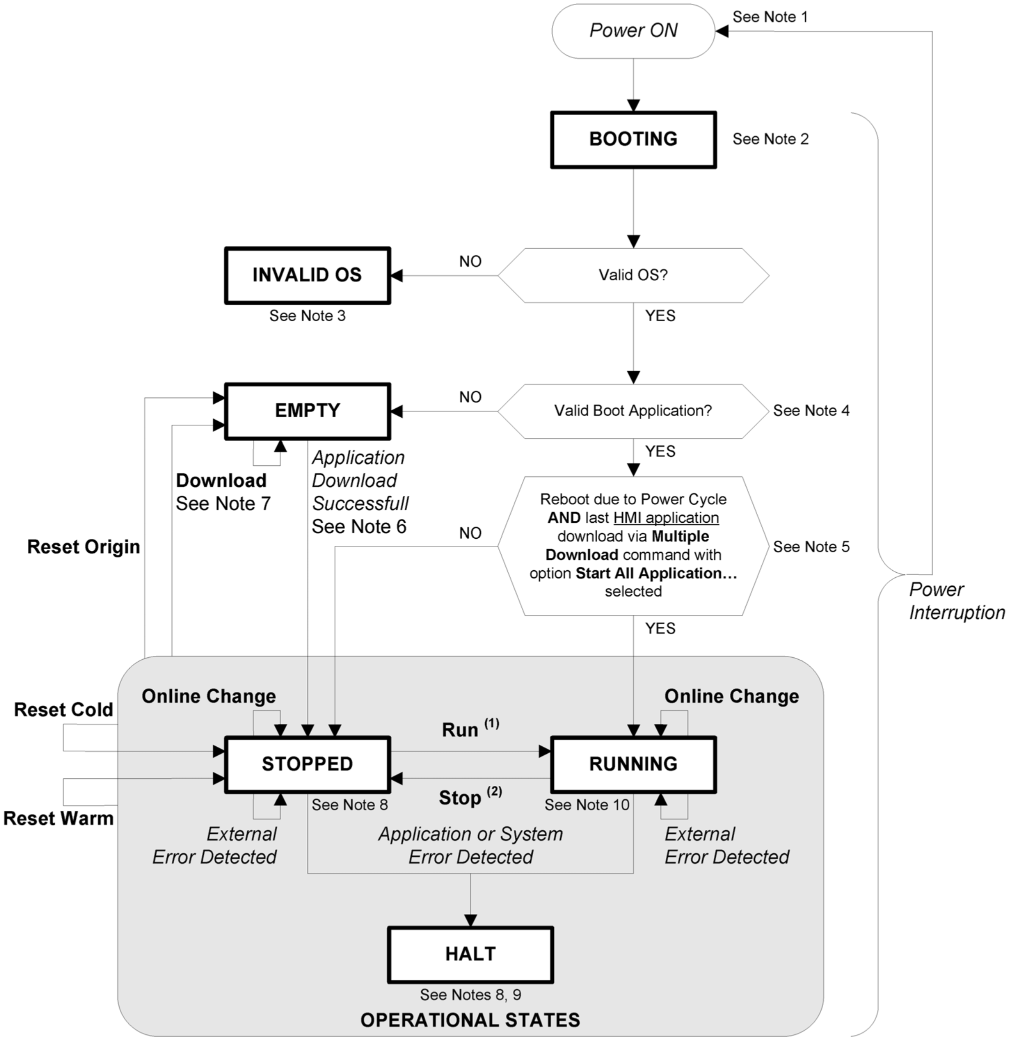
This chapter contains the following sections:

6.1 Controller State Diagram

Controller State Diagram

Controller State Diagram

The following diagram describes the controller operating mode:



Legend:

- Controller states are indicated in **ALL-CAPS BOLD**
- User and application commands are indicated in **Bold**
- System events are indicated in *Italics*
- Decisions, decision results and general information are indicated in normal text

[1] For details on STOPPED to RUNNING state transition, refer to Run Command *(see page 59)*.

[2] For details on RUNNING to STOPPED state transition, refer to Stop Command *(see page 59)*.

Note 1

The Power Cycle (Power Interruption followed by a Power ON) deletes all output forcing settings. Refer to Controller State and Output Behavior *(see page 56)* for further details.

Note 2

The outputs will assume their initialization states.

Note 3

HMI download screen is displayed prompting the user to download the firmware, HMI and Control application.

Note 4

The application is loaded into RAM after verification of a valid Boot application.

Note 5

The state of the controller will be RUNNING after a reboot if the reboot was provoked by a Power Cycle and the **HMI application** had been downloaded using a **Multiple Download...** command with option **Start all application after download or online change** selected.

Note 6

During a successful application download the following events occur:

- The application is loaded directly into RAM.
- By default, the Boot application is created and saved into the Flash memory.

Note 7

However, there are two important considerations in this regard:

- **Online Change:** An online change (partial download) initiated while the controller is in the RUNNING state returns the controller to the RUNNING state if successful.
  Before using the **Login with online change** option, test the changes to your application program in a virtual or non-production environment and confirm that the controller and attached equipment assume their expected conditions in the RUNNING state.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Always verify that online changes to a RUNNING application program operate as expected before downloading them to controllers.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

NOTE: Online changes to your program are not automatically written to the Boot application, and will be overwritten by the existing Boot application at the next reboot. If you wish your changes to persist through a reboot, manually update the Boot application by selecting **Create boot application** in the Online menu.

- **Multiple Download:** SoMachine has a feature that allows you to perform a full application download to multiple targets on your network or fieldbus.
  One of the default options when you select the **Multiple Download...** command is the **Start all applications after download or online change** option, which restarts all download targets in the RUNNING state, irrespective of their last controller state before the multiple download was initiated. Deselect this option if you do not want all targeted controllers to restart in the RUNNING state.
  In addition, before using the **Multiple Download...** option, test the changes to your application program in a virtual or non-production environment and confirm that the targeted controllers and attached equipment assume their expected conditions in the RUNNING state.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Always verify that your application program will operate as expected for all targeted controllers and equipment before issuing the "**Multiple Download…**" command with the "**Start all applications after download or online change**" option selected.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Note 8

The SoMachine software platform allows many powerful options for managing task execution and output conditions while the controller is in the STOPPED or HALT states. Refer to Controller State and Output Behavior *(see page 55)* for further details.

Note 9

To exit the HALT state it is necessary to issue one of the Reset commands (Reset Warm, Reset Cold, Reset Origin), download an application or cycle power.

Note 10

The RUNNING state has two exceptional conditions that will be indicated in run state or error messages on HMI screen.

- RUNNING with External Error: You may exit this exceptional condition by clearing the external error. No controller commands are required.
- RUNNING with Breakpoint: Refer to Controller State Description *(see page 52)* for further details on this exceptional condition.

6.2 Controller States Description

Controller States Description

Introduction

This section provides a detailed description of the controller states.

⚠ **WARNING**

UNINTENDED EQUIPMENT OPERATION

- Never assume that your controller is in a certain controller state before commanding a change of state, configuring your controller options, uploading a program, or modifying the physical configuration of the controller and its connected equipment.
- Before performing any of these operations, consider the effect on all connected equipment.
- Before acting on a controller, always positively confirm the controller state by checking for the presence of output forcing, and reviewing the controller status information via SoMachine [1].

Failure to follow these instructions can result in death, serious injury, or equipment damage.

[1] **Note:** The controller states can be read in the PLC_R.i_wStatus system variable of the XBT PLCSystem library *(see Magelis XBTGC, XBTGT, XBTGK HMI Controller, System Functions and Variables, XBT PLCSystem Library Guide)*

Controller States Table

The following table describes the controller states:

Controller State	Description
BOOTING	The controller executes the boot firmware and its own internal self-tests. It then checks the checksum of the firmware and user applications. It does not execute the application nor does it communicate.
INVALID_OS	There is not a valid firmware file present In the Flash memory. The controller does not execute the application. Communication is only possible through the USB host port, and then only for uploading a valid OS.
EMPTY	There is no application in memory or the application is invalid.
RUNNING	The controller is executing a valid application.

Controller State	Description
RUNNING with Breakpoint	This state is the same as the RUNNING state with the following exceptions: ● The task-processing portion of the program does not resume until the breakpoint is cleared. See CoDeSys online help in SoMachine for details on breakpoints management.
RUNNING with detection of an *External Error*	This state is the same as the normal RUNNING state.
STOPPED	The controller has a valid application that is stopped. See Details of the STOPPED State *(see page 53)* for an explanation of the behavior of outputs and field buses in this state.
STOPPED with detection of an *External Error*	This state is the same as the normal STOPPED state.
HALT	The controller stops executing the application because it has detected an Application or a System Error. This description is the same as for the STOPPED state with the following exceptions: ● The task responsible for the Application Error always behaves as if the **Update IO while in stop** option was not selected. All other tasks follow the actual setting.

Details of the STOPPED State

The following statements are always true for the STOPPED state:

- Ethernet, Serial (Modbus, ASCII, etc.), and USB communication services remain operational and commands written by these services can continue to affect the application, the controller state, and the memory variables.
- All outputs initially assume their configured state (**Keep current values** or **Set all outputs to default**) or the state dictated by output forcing if used. The subsequent state of the outputs depends on the value of the **Update IO while in stop** setting and on commands received from remote devices.

NOTE: There are no local or remote I/O on XBT GT and XBT GK HMI Controllers. %I input memory variables and %Q output memory variables are attached to CANopen data if configured.

Task and I/O Behavior When Update IO While In Stop Is Selected

When the **Update IO while in stop** setting is selected:

- The Read Inputs operation continues normally. The physical inputs are read and then written to the %I input memory variable.
- The Task Processing operation is not executed.

- The Write Outputs operation continues. The %Q output memory variable is updated to reflect either the **Keep current values** configuration or the **Set all outputs to default** configuration, adjusted for any output forcing, and then written to the physical outputs.

NOTE: Commands received by Ethernet, Serial, USB, and CAN communications can continue to write to the memory variables. Changes to the %Q output memory variables are written to the physical outputs.

CAN Behavior When Update IO While In Stop Is Selected

The following is true for the CAN buses when the Update IO while in stop setting is selected:

- The CAN bus remains fully operational. Devices on the CAN bus continue to perceive the presence of a functional CAN Master.
- TPDO and RPDO continue to be exchanged.
- The optional SDO, if configured, continue to be exchanged.
- The Heartbeat and Node Guarding functions, if configured, continue to operate.
- If the **Behaviour for outputs in Stop** field is set to **Keep current values**, the TPDOs continue to be issued with the last actual values.
- If the **Behaviour for outputs in Stop** field is **Set all outputs to default** the last actual values are updated to the default values and subsequent TPDOs are issued with these default values.

Task and I/O Behavior When Update IO While In Stop Is Not Selected

When the **Update IO while in stop** setting is not selected, the controller sets the I/O to either the **Keep current values** or **Set all outputs to default** condition (as adjusted for output forcing if used). After this, the following becomes true:

- The Read Inputs operation ceases. The %I input memory variable is frozen at its last values.
- The Task Processing operation is not executed.
- The Write Outputs operation ceases. The %Q output memory variables can be updated via the Ethernet, Serial, and USB connections. However, the physical outputs are unaffected and retain the state specified by the configuration options.

CAN Behavior When Update IO While In Stop Is Not Selected

The following is true for the CAN buses when the **Update IO while in stop** setting is not selected:

- The CAN Master ceases communications. Devices on the CAN bus assume their configured fallback states.
- TPDO and RPDO exchanges cease.
- Optional SDO, if configured, exchanges cease.
- The Heartbeat and Node Guarding functions, if configured, stop.
- The current or default values, as appropriate, are written to the TPDOs and sent once before stopping the CAN Master.

6.3 State Transitions and System Events

Overview

This section begins with an explanation of the output states possible for the controller. It then presents the system commands used to transition between controller states and the system events that can also affect these states. It concludes with an explanation of the Remanent variables, and the circumstances under which different variables and data types are retained through state transitions.

What's in this Section?

This section contains the following topics:

Controller States and Output Behavior

Introduction

The XBT GT/GK HMI Controller defines output behavior in response to commands and system events in a way that allows for greater flexibility. An understanding of this behavior is necessary before discussing the commands and events that affect controller states. For example, typical controllers define only two options for output behavior in stop: fallback to default value or keep current value.

The possible output behaviors and the controller states to which they apply are:

- ControllerLockout Feature
- Managed by Application Program
- Keep Current Values
- Set All Outputs to Default
- Output Forcing

ControllerLockout Feature

The **ControllerLockout** feature locks or unlocks the controller stop mode. A locked controller cannot be restarted until the controller is unlocked.

Attempts to restart a locked controller are ignored and a message appears.You can only initiate lockout once the controller is in STOPPED state. If the controller is in RUNNING state and you attempt to lockout, the attempt is ignored and a message appears.

The **ControllerLockout** is not managed through SoMachine, it is an internal boolean variable (_ControllerLockout) of the HMI in Vijeo Designer.

For more information on managing this variable, refer to the Vijeo Designer Online Help.

Managed by Application Program

Your application program manages outputs normally. This applies in the RUNNING and RUNNING with External Error states.

Keep Current Values

You can select this option by choosing **Keep current values** in the **Behaviour for outputs in Stop** dropdown menu of the **PLC Settings** sub-tab of the **Controller Editor**. To access the Controller Editor, right-click on the controller in the device tree and select **Edit Object**.

This output behavior applies in the STOPPED and HALT controller states. Outputs are set to and maintained in their current state, although the details of the output behavior varies greatly depending on the setting of the **Update IO while in stop** option and the actions commanded via configured fieldbuses. Refer to Controller States Description *(see page 52)* for more details on these variations.

Set All Outputs to Default

You can select this option by choosing **Set all outputs to default** in the **Behaviour for outputs in Stop** dropdown menu of the **PLC Settings** sub-tab of the **Controller Editor**. To access the **Controller Editor**, right-click on the controller in the device tree and select **Edit Object**.

This output behavior applies in the STOPPED and HALT controller states. Outputs are set to their user-defined default values, although the details of the output behavior varies greatly depending on the setting of the **Update IO while in stop** option and the actions commanded via configured fieldbuses. Refer to Controller States Description *(see page 52)* for more details on these variations.

Output Forcing

The controller allows you to force the state of selected outputs to a defined value for the purposes of system testing, commissioning and maintenance.

You are only able to force the value of an output while your controller is connected to SoMachine.

To do so you use the Force Values command in the Debug/Watch menu.

Output forcing overrides all other commands to an output irrespective of the task programming that is being executed.

When you logout of SoMachine when output forcing has been defined, you are presented with the option to retain output forcing settings. If you select this option, the output forcing continues to control the state of the selected outputs until you download an application or use one of the Reset commands.

When the option **Update IO while in stop**, if supported by your controller, is checked (default state), the forced outputs keep the forcing value even when the logic controller is in STOP.

Output Forcing Considerations

The output you wish to force must be contained in a task that is currently being executed by the controller. Forcing outputs in unexecuted tasks, or in tasks whose execution is delayed either by priorities or events will have no effect on the output. However, once the task that had been delayed is executed, the forcing will take effect at that time.

Depending on task execution, the forcing could impact your application in ways that may not be obvious to you. For example, an event task could turn on an output. Later, you may attempt to turn off that output but the event is not being triggered at the time. This would have the effect of the forcing apparently being ignored. Further, at a later time, the event could trigger the task at which point the forcing would take effect.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- You must have a thorough understanding of how forcing will affect the outputs relative to the tasks being executed.
- Do not attempt to force I/O that is containted in tasks that you are not certain will be executed in a timely manner, unless your intent is for the forcing to take affect at the next execution of the task whenever that may be.
- If you force an output and there is no apparent affect on the physical output, do not exit SoMachine without removing the forcing.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Commanding State Transitions

Run Command

Effect: Commands a transition to the RUNNING controller state.

Starting Conditions: BOOTING or STOPPED state.

Methods for Issuing a Run Command:

- SoMachine Online Menu: Select the **Start** command.
- By an HMI command using the PLC_W. q_wPLCControl and PLC_W. q_uiOpenPLCControl system variables of the XBT PLCSystem library *(see Magelis XBTGC, XBTGT, XBTGK HMI Controller, System Functions and Variables, XBT PLCSystem Library Guide)*.
- **Login with online change** option: An online change (partial download) initiated while the controller is in the RUNNING state returns the controller to the RUNNING state if successful.
- **Multiple Download** Command: sets the controllers into the RUNNING state if the **Start all applications after download or online change** option is selected, irrespective of whether the targeted controllers were initially in the RUNNING, STOPPED, HALT or EMPTY state.
- The controller is restarted into the RUNNING state automatically under certain conditions.

Refer to Controller State Diagram *(see page 48)* for further details.

Stop Command

Effect: Commands a transition to the STOPPED controller state.

Starting Conditions: BOOTING, EMPTY or RUNNING state.

Methods for Issuing a Run Command:

- SoMachine Online Menu: Select the **Stop** command.
- By an internal call by the application or an HMI command using the PLC_W. q_wPLCControl and PLC_W. q_uiOpenPLCControl system variables of the XBT PLCSystem library *(see Magelis XBTGC, XBTGT, XBTGK HMI Controller, System Functions and Variables, XBT PLCSystem Library Guide)*.
- **Login with online change** option: An online change (partial download) initiated while the controller is in the STOPPED state returns the controller to the STOPPED state if successful.
- **Download** Command: implicitly sets the controller into the STOPPED state.
- **Multiple Download** Command: sets the controllers into the STOPPED state if the **Start all applications after download or online change** option is not selected, irrespective of whether the targeted controllers were initially in the RUNNING, STOPPED, HALT or EMPTY state.

- REBOOT by Script: The file transfer script on a USB memory key can issue a REBOOT as its final command. The controller will be rebooted into the STOPPED state provided the other conditions of the boot sequence allow this to occur. Refer to Saving your Application and Firmware on a USB Memory Key *(see page 93)* and Reboot *(see page 93)* for further details.
- The controller is restarted into the STOPPED state automatically under certain conditions.

Refer to Controller State Diagram *(see page 48)* for further details.

Reset Warm

Effect: Resets all variables, except for the remanent variables, to their default values. Places the controller into the STOPPED state.

Starting Conditions:
- RUNNING, STOPPED, or HALT states.
- ControllerLockout = 0.

Methods for Issuing a Reset Warm Command:
- SoMachine Online Menu: Select the **Reset warm** command.
- By an internal call by the application or an HMI command using the PLC_W. q_wPLCControl and PLC_W. q_uiOpenPLCControl system variables of the XBT PLCSystem library *(see Magelis XBTGC, XBTGT, XBTGK HMI Controller, System Functions and Variables, XBT PLCSystem Library Guide)*.

Effects of the Reset Warm Command:
1. The application stops.
2. Forcing is erased.
3. Diagnostic indications for detected errors are reset.
4. The values of the retain variables are maintained.
5. The values of the retain-persistent variables are maintained.
6. All non-located and non-remanent variables are reset to their initialization values.
7. All fieldbus communications are stopped and then restarted after the reset is complete.
8. All I/O are briefly reset to their initialization values and then to their user-configured default values.

For details on variables, refer to Remanent Variables *(see page 67)*.

Reset Cold

Effect: Resets all variables, except for the retain-persistent type of remanent variables, to their initialization values. Places the controller into the STOPPED state.

Starting Conditions:
- RUNNING, STOPPED, or HALT states.
- ControllerLockout = 0.

Methods for Issuing a Reset Cold Command:

- SoMachine Online Menu: Select the **Reset cold** command.
- By an internal call by the application or an HMI command using the PLC_W. q_wPLCControl and PLC_W. q_uiOpenPLCControl system variables of the XBT PLCSystem library *(see Magelis XBTGC, XBTGT, XBTGK HMI Controller, System Functions and Variables, XBT PLCSystem Library Guide)*.

Effects of the Reset Cold Command:

1. The application stops.
2. Forcing is erased.
3. Diagnostic indications for detected errors are reset.
4. The values of the retain variables are reset to their initialization value.
5. The values of the retain-persistent variables are maintained.
6. All non-located and non-remanent variables are reset to their initialization values.
7. All fieldbus communications are stopped and then restarted after the reset is complete.
8. All I/O are briefly reset to their initialization values and then to their user-configured default values.

For details on variables, refer to Remanent Variables *(see page 67)*.

Reset Origin

Effect: Resets all variables, including the remanent variables, to their initialization values. Erases all user files on the controller. Places the controller into the EMPTY state.

Starting Conditions:

- RUNNING, STOPPED, or HALT states.
- ControllerLockout = 0.

Methods for Issuing a Reset Origin Command:

- SoMachine Online Menu: Select the **Reset origin** command.

Effects of the Reset Origin Command:

1. The application stops.
2. Forcing is erased.
3. All user files (Boot application, data logging) are erased.
4. Diagnostic indications for detected errors are reset.
5. The values of the retain variables are reset.
6. The values of the retain-persistent variables are reset.
7. All non-located and non-remanent variables are reset.
8. All fieldbus communications are stopped.
9. All I/O are reset to their initialization values.

For details on variables, refer to Remanent Variables *(see page 67)*.

Reboot

Effect: Commands a reboot of the controller.

Starting Conditions:

- Any state.
- ControllerLockout = 0.

Methods for Issuing the Reboot Command:

- Power cycle.
- REBOOT by Script: The file transfer script on a USB memory key issues a REBOOT as its final command. The controller will be rebooted into the STOPPED state provided the other conditions of the boot sequence allow this to occur. Refer to Saving your Application and Firmware on a USB Memory Key *(see page 93)* for further details.

Effects of the Reboot:

1. The state of the controller depends on a number of conditions:
   a. The controller state will be RUNNING if:
   - The Reboot was provoked by a power cycle, and
   - The Reboot was provoked by a power cycle and the HMI application had been downloaded using a **Multiple Download** command with option **Start all application after download** or online change selected.
   - The Reboot was provoked by an HMI application download using a **Multiple Download** command with option **Start all application after download** or online change selected.

   b. The controller state will be STOPPED if:
   - Controller state was STOPPED prior to a power cycle, or
   - The Reboot was provoked by a power cycle and the HMI application had been downloaded using a **Multiple Download** command with option **Start all application after download** or online change not selected.
   - The Reboot was provoked by an HMI application download using a **Multiple Download** command with option **Start all application after download** or online change not selected.

   c. The controller state will be EMPTY if there is no boot application or the boot application is invalid.

   d. The controller state will be INVALID_OS if there is no valid OS.
2. Forcing is maintained if the boot application is loaded successfully. If not, forcing is erased.
3. Diagnostic indications for detected errors are reset.
4. The values of the retain variables are restored if saved context is valid.
5. The values of the retain-persistent variables are restored if saved context is valid.
6. All non-located and non-remanent variables are reset to their initialization values.

7. All fieldbus communications are stopped and restarted after the boot application is loaded successfully.
8. All I/O are reset to their initialization values and then to their user-configured default values if the controller assumes a STOPPED state after the reboot.

For details on variables, refer to Remanent Variables *(see page 67)*.

NOTE: The Check context test concludes that the context is valid when the application and the remanent variables are the same as defined in the Boot application.

NOTE: If you make an online change to your application program while your controller is in the RUNNING or STOPPED state but do not manually update your Boot application, the controller will detect a difference in context at the next reboot, the remanent variables will be reset as per a Reset cold command, and the controller will enter the STOPPED state.

Download Application

Effect: Loads your application executable into the RAM memory. Optionally, creates a Boot application in the Flash memory.

Starting Conditions:
- RUNNING, STOPPED, HALT, and EMPTY states.
- ControllerLockout = 0.

Methods for Issuing the Download Application Command:
- SoMachine:
  Two options exist for downloading a full application:
  - Download command.
  - Multiple Download command.

  For important information on the application download commands, refer to Controller State Diagram *(see page 48)*.
- FTP: Load Boot application file to the Flash memory using FTP. The updated file is applied at the next reboot.
- USB memory key: Load Boot application file using a USB memory key connected to the controller USB host port. The updated file is applied at the next reboot. Refer to Saving your Application and Firmware on a USB Memory Key *(see page 93)* for further details.

Effects of the SoMachine Download Command:
1. The existing application stops and then is erased.
2. If valid, the new application is loaded and the controller assumes a STOPPED state.
3. Forcing is erased.
4. Diagnostic indications for detected errors are reset.
5. The values of the retain variables are reset to their initialization values.
6. The values of any existing retain-persistent variables are maintained.
7. All non-located and non-remanent variables are reset to their initialization values.

8. All fieldbus communications are stopped and then any configured fieldbus of the new application is started after the download is complete.
9. All I/O are reset to their initialization values and then set to the new user-configured default values after the download is complete.

For details on variables, refer to Remanent Variables *(see page 67)*.

Effects of the FTP or USB memory key Download Command:

There are no effect until the next reboot. At the next reboot, the effects are the same as a reboot with an invalid context. Refer to Reboot *(see page 93)*.

Error Detection, Types, and Management

Detected Error Management

The controller manages 3 types of detected errors:

- external detected errors
- application detected errors
- system detected errors

The following table describes the types of errors that may be detected:

Type of Error Detected	Description	Resulting Controller State
External Error Detected	External errors are detected by the system while RUNNING or STOPPED but do not affect the ongoing controller state. An external error is detected in the following cases: ● A connected device reports an error to the controller ● The controller detects an error with an external device whether or not it reports an error, for example when the external device is communicating but not properly configured for use with the controller ● The controller detects an error with the state of an output ● The controller detects a loss of communication with a device ● The controller is configured for a module that is not present or not detected ● The boot application in Fash memory is not the same as the one in RAM. Examples: ● output short circuit ● missing expansion module ● communication lost ● etc.	RUNNING with External Error Detected Or STOPPED with External Error Detected
Application Error Detected	An application error is detected when improper programming is encountered or when a task watchdog threshold is exceeded. Examples: ● task (software) watchdog exception ● execution of an unknown function ● etc.	HALT

Type of Error Detected	Description	Resulting Controller State
System Error Detected	A system error is detected when the controller enters a condition that cannot be managed during runtime. Most such conditions result from firmware or hardware exceptions, but there are some cases when incorrect programming can result in the detection of a system error, for example, when attempting to write to memory that was reserved during runtime. Examples: ● exceeding the defined size of an array ● etc.	BOOTING → EMPTY

NOTE: Refer to the XBT PLCSystem library *(see Magelis XBTGC, XBTGT, XBTGK HMI Controller, System Functions and Variables, XBT PLCSystem Library Guide)* for more detailed information on diagnostics.

NOTE: For XBT GT/GK HMI Controller, System (hardware) watchdog overflow detection is not supported.

Remanent Variables

Remanent Variables

Remanent variables can retain their values in the event of power outages, reboots, resets, and application program downloads. There are multiple types of remanent variables, declared individually as "retain" or "persistent", or in combination as "retain-persistent".

NOTE: For this controller, variables declared as persistent have the same behavior as variables declared as retain-persistent.

The following table describes the behavior of remanent variables in each case:

Action	VAR	VAR RETAIN	VAR PERSISTENT and RETAIN-PERSISTENT
Online change to application program	X	X	X
Stop	X	X	X
Power cycle	-	X	X
Reset warm	-	X	X
Reset cold	-	-	X
Reset origin	-	-	-
Download of application program	-	-	X
X The value is maintained **-** The value is reinitialized			

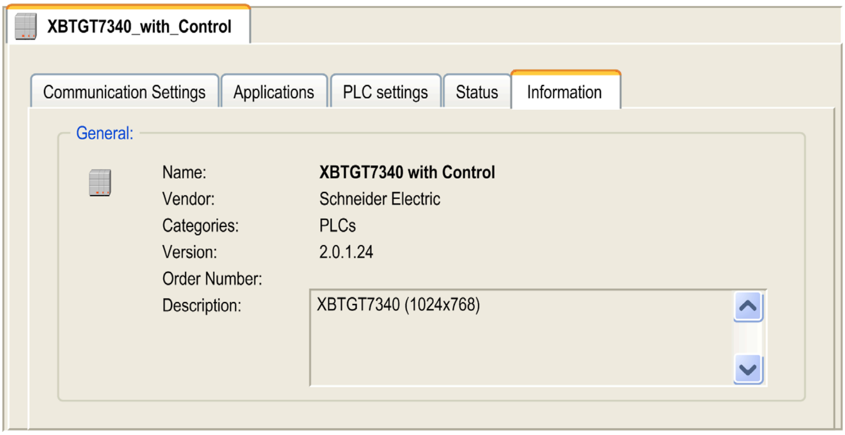
Controller Configuration

7

Device Editor

Introduction

Configure and monitor your XBT GT/GK HMI Controller using the **Device Editor**. The following screen-shot shows the **Information** tab of the **Device Editor** window:



Name: XBTGT7340 with Control
Vendor: Schneider Electric
Categories: PLCs
Version: 2.0.1.24
Order Number:
Description: XBTGT7340 (1024x768)

For more detailed information refer to the CoDeSys Online-Help.

XBT GT/GK HMI Controller Device Editor Window

To open the XBT GT/GK HMI Controller Device Editor, **double click** on the controller name *(see page 18)* or **right click** → **Edit Object**.

NOTE: You can also access the Device Editor window from the *graphical configuration (see SoMachine, Programming Guide)* editor.

Tabs Description

The following table provides a description of the tabs available from the Device Editor window:

Tab	Description
Communication Settings	Allows configuring the connection between the programming system and the controller (not available for expansion modules).
Applications	Shows the applications currently running on the controller and allows removing applications from the controller (not available for expansion modules).
PLC Settings	Allows configuring the fallback of the outputs.
Status	Displays device-specific status and diagnostic messages.
Information	Displays general information about the device (name, description, provider, version, image).

NOTE: For more detailed information refer to the CoDeSys Online-Help.

Ethernet Configuration

8

IP Address Configuration

Introduction

Setting up an Ethernet connection and IP address configuration with the HMI controllers is done using Vijeo-Designer.

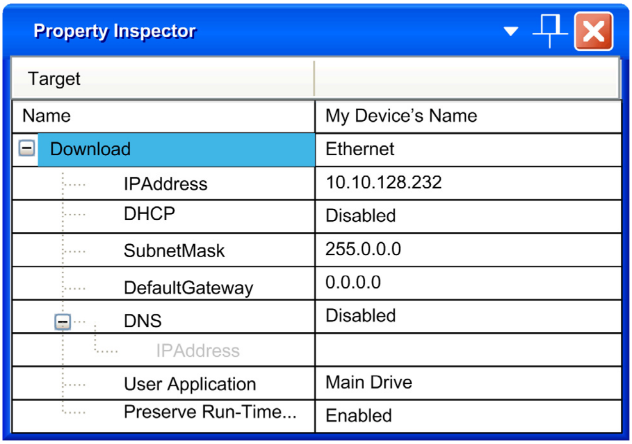
There are two ways to assign the IP address of the controller with Vijeo-Designer:

- DHCP server
- Fixed IP address

NOTE: If the above addressing modes are not operational, the PLC starts with a *default IP address (see page 72)* derived from its MAC address.

Ethernet Configuration

For the HMI controller, the Ethernet configuration is done via the Vijeo-Designer **Property Inspector** window, as shown in the next figure:



Target	
Name	My Device’s Name
Download	Ethernet
IPAddress	10.10.128.232
DHCP	Disabled
SubnetMask	255.0.0.0
DefaultGateway	0.0.0.0
DNS	Disabled
IPAddress	
User Application	Main Drive
Preserve Run-Time...	Enabled

NOTE: The Ethernet configuration parameters are applied after a download of the HMI application.

The following table briefly explains the different parameters needed for setting up an Ethernet configuration:

Element	Description
Download	Choose the project download method you want in the drop down menu list. When configuring ethernet connection, select **Ethernet**. The project download methods are: ● Ethernet ● File System ● USB ● SoMachine
IP Address	IP address of the controller.
DHCP	When DHCP is: ● **Enable**: The controller automatically retrieves an IP address from a DHCP server. ● **Disabled**: The controller uses a static IP address.
SubnetMask	When using a static IP setting, provide the subnet mask of your network.
DefaultGateway	When using a static IP setting, provide the default gateway of your network.
DNS	**Enable** DNS to use domain names instead of IP addresses.
DNS IP Address	When using DNS, provide the IP address for the DNS server.

NOTE: For more information on how to configure the Ethernet connection between your computer and the HMI controller, refer to Vijeo-Designer online help.

Default IP Address

The default IP address is based on MAC address of the device. The first two bytes are 10 and 10. The last two bytes are the last two bytes of the device’s MAC address.

The default subnet mask is 255.0.0.0.

NOTE: A MAC address has an hexadecimal format and an IP address has a decimal format. Convert the MAC address into decimal format.

Example: If the MAC address is 00.80.F4.01.80.F2, the default IP address is 10.10.128.242.

CANopen Configuration

9

Introduction

This chapter describes how to configure the CANopen network interface of the XBT GT/GK HMI Controller.

What's in this Chapter?

This chapter contains the following topics:

CANopen Interface Configuration

XBT GT/GK HMI Controller Maximum Hardware Configuration

Up to 10 CANopen remote devices can be connected to the CANopen Master Unit.

XBT GT/GK HMI Controller Software Requirements

The maximum number of Received PDO RPDO is 20.

The maximum number of Transmitted PDO TPDO is 20.

Adding the CANopen Expansion Modules

When adding a CANopen expansion module (XBT ZGCANM or XBT ZGCANMS0) to the XBT GT/GK HMI Controller, the CANbus node is automatically created. Additional CANopen devices can be added to the manager.

⚠ **WARNING**

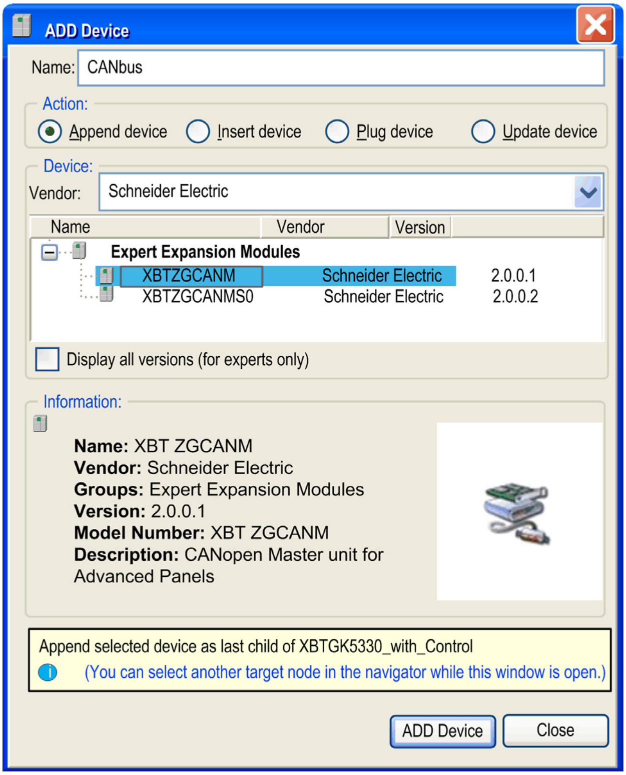
UNINTENDED EQUIPMENT OPERATION

- Only use software approved by Schneider Electric for use with this equipment.
- Update your application program every time you change the physical hardware configuration.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

The following procedure explains how to add the CANopen expansion module to your project:

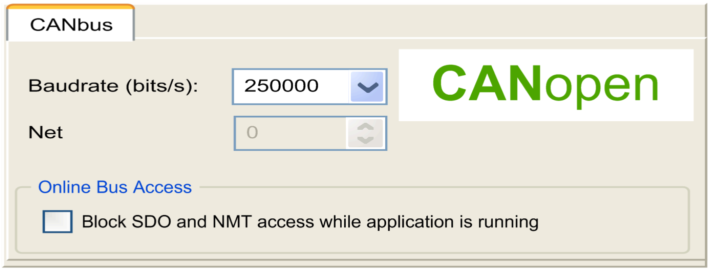
Step	Action
1	Right-click the XBT GT/GK HMI Controller node in the **Devices** window and select **Add Device**. **Tip:** Alternatively, click **Project** → **Add Devices** from the main menu bar.

Step	Action
2	In the **Add Device** window, select the CANopen expansion module (XBT ZGCANM or XBT ZGCANMS0), as shown in the figure below:  ADD Device Name: CANbus Action: ● Append device ○ Insert device ○ Plug device ○ Update device Device: Vendor: Schneider Electric Name · Vendor · Version Expert Expansion Modules XBTZGCANM · Schneider Electric · 2.0.0.1 XBTZGCANMS0 · Schneider Electric · 2.0.0.2 □ Display all versions (for experts only) Information: **Name:** XBT ZGCANM **Vendor:** Schneider Electric **Groups:** Expert Expansion Modules **Version:** 2.0.0.1 **Model Number:** XBT ZGCANM **Description:** CANopen Master unit for Advanced Panels Append selected device as last child of XBTGK5330_with_Control (You can select another target node in the navigator while this window is open.) ADD Device · Close **Note**: To sort the devices by type in the **Add Device** window, select **Schneider Electric** in the **Vendor** list box.
3	Rename the CANopen expansion module by typing a name in the **Name** field. **Note:** ● Do not use spaces or special characters (%, #). ● The length of the name cannot exceed 32 characters.
4	Click **Add Device** to add the device to your project. **Result:** The **Device** window is updated with a CANbus node and its associated CANopen_Manager. The **Add Device** window opens again. **NOTE:** You cannot physically add any I/O expansion modules with a CANopen module, refer to XBT GT/GK HMI Controller *Maximum Hardware Configuration (see page 74)* for additional information.

NOTE: You can also add a CANopen Master Unit using the *Graphical Configuration Editor (see SoMachine, Programming Guide)*. Refer to *Adding Expansion Modules (see SoMachine, Programming Guide)* for additional information.

Baudrate Configuration

The following table provides the procedure for accessing the CANopen Baudrate configuration screen:

Step	Action
1	Double click the **CANbus** node in the **Devices** window. **Result:** The **CANbus** configuration screen appears:  CANbus Baudrate (bits/s): 250000 Net 0 Online Bus Access □ Block SDO and NMT access while application is running
2	Select the **CANbus** tab.
3	Configure the baudrate using the **Baudrate (bits/s)** menu list. By default, the value is set to 250,000 bit/s.
4	Configure the net using the **Net** menu list. By default, the value is set to 0.
5	Configure the online bus access by clicking **Block SDO and NMT access while application is running**. By default, the online bus access is activated.

NOTE: You can also configure the baudrate using the *Graphical Configuration Editor (see SoMachine, Programming Guide)*. Refer to *Configuring Device Parameters (see SoMachine, Programming Guide)* for additional information.

CANopen Network Manager

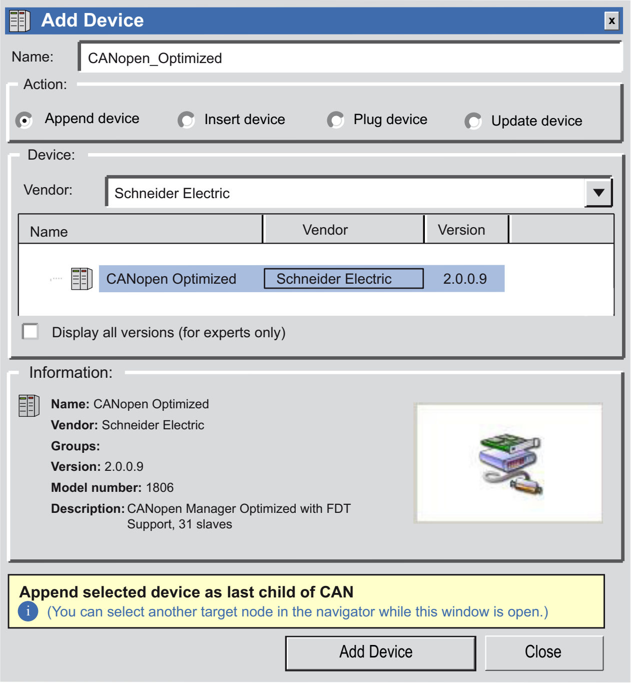
Configure the CANopen **Network_Manager** when using CANopen:

Element	Description
CANopen_Optimized- Network_Manager	Used to support the CANbus configuration by internal functions [1].
Legend	
1	Refer to *CANopen Optimized Manager (see page 77)* for additional information on the configuration.

CANopen Optimized Manager

Adding the CANopen Optimized Manager

Add the **CANopen_Optimized** Manager to your project to attach a remote device. To add the **CANopen_Optimized** Manager follow the steps:

Step	Action
1	Right-click on the **CANbus** node in the **Devices** window and choose **Add Device....** **Result:** The **Add Device** windows opens:  Add Device Name: CANopen_Optimized Action: ● Append device ○ Insert device ○ Plug device ○ Update device Device: Vendor: Schneider Electric Name Vendor Version CANopen Optimized Schneider Electric 2.0.0.9 □ Display all versions (for experts only) Information: **Name:** CANopen Optimized **Vendor:** Schneider Electric **Groups:** **Version:** 2.0.0.9 **Model number:** 1806 **Description:** CANopen Manager Optimized with FDT Support, 31 slaves **Append selected device as last child of CAN** (You can select another target node in the navigator while this window is open.)
2	Choose one of the **CANopen_Optimized** manager selections available.
3	Click the **Add Device** button. **Result:** The **CANopen_Optimized** manager is added to the project tree in the **Devices** window.

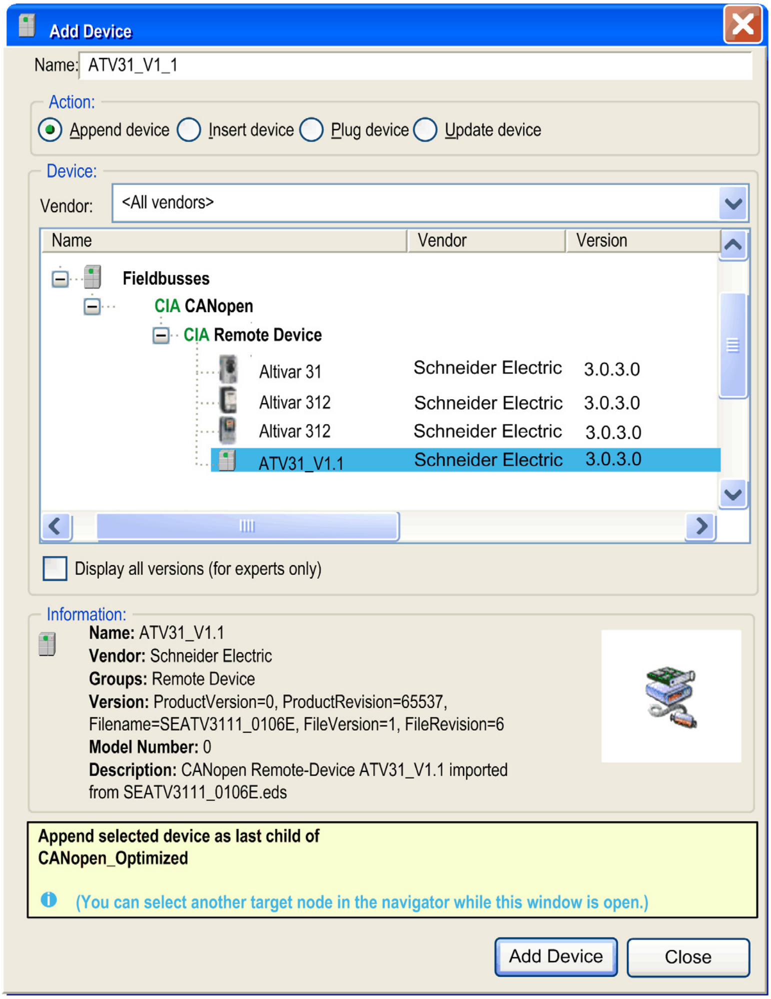
CANopen Optimized Manager Configuration Screen

You can access the **CANopen Optimized** Manager configuration screen by double clicking the CANopen_Optimized node from the **Device** window. Refer to the CANopen Manager part from the CoDeSys Online-Help for additional information.

CANopen Remote Devices

Adding a Remote Device to the CANopen_Optimized manager

To add a remote device to the **CANopen_Optimized** manager, follow the steps in the table below:

Step	Action
1	Right click on the **CANopen_Optimized** manager node and choose **Add Device**.
2	In the **Add Device** window, select **All Vendor** in the **Vendor** list box, as shown in the figure below:  Add Device Name: ATV31_V1_1 Action: ● Append device ○ Insert device ○ Plug device ○ Update device Device: Vendor: <All vendors> Name Vendor Version Fieldbusses CIA CANopen CIA Remote Device Altivar 31 Schneider Electric 3.0.3.0 Altivar 312 Schneider Electric 3.0.3.0 Altivar 312 Schneider Electric 3.0.3.0 ATV31_V1.1 Schneider Electric 3.0.3.0 □ Display all versions (for experts only) Information: **Name:** ATV31_V1.1 **Vendor:** Schneider Electric **Groups:** Remote Device **Version:** ProductVersion=0, ProductRevision=65537, Filename=SEATV3111_0106E, FileVersion=1, FileRevision=6 **Model Number:** 0 **Description:** CANopen Remote-Device ATV31_V1.1 imported from SEATV3111_0106E.eds Append selected device as last child of CANopen_Optimized (You can select another target node in the navigator while this window is open.) Add Device Close

Step	Action
3	Select the remote device you want to add.
4	Rename your device by typing a name in the **Name** field. **Note:** ● Do not use spaces or special characters (%, #). ● The name length cannot exceed 32 characters.
5	Click **Add Device** to add the device to your project. **Result:** The **Device** window is refreshed with the new remote devices associated to the CANopen_Manager. The **Add Device** window opens again. You can then: ● repeat step 3 to add another remote device[1], or ● click **Close**. [1] Up to 16 CANopen remote devices *(see Magelis XBTGC HMI Controller, Programming Guide)* can be connected to the CANopen Master Unit.

CANopen Remote Device Configuration Screen

You can access the remote device configuration screen by double clicking on the device from the **Devices** screen. Refer to CANopen Remote Device part from the CoDeSys Online-Help for additonal information.

Remote Devices Available with CANopen

The following list shows the remote devices available with CANopen and supported by SoMachine:

- Variable speed drives such as ATV 31 or ATV 71.
- Servo drives such as Lexium 05.
- Integrated drives such as ILA1F, ILE1F or the ILS1F.
- Opto-electronic encoders such as the Osicoder.
- Configurable safety controllers such as the Preventa XPSMC••ZC.
- Stepper motor drives such as SD328.
- Motor management and protection systems such as TeSysT.
- Starter controllers such as TeSysU.
- Distributed I/Os such as FTB or TVD_OTB.

NOTE: Other CANopen devices can be added using their electronic data sheet (EDS) files.

Refer to *Supported Devices (see SoMachine, Introduction)* for additional information.

For additional information on these remote devices refer to external devices documentation available on Schneider Electric Website.

Serial Line Configuration

10

Introduction

This chapter describes how to configure the serial line communication of the XBT GT/GK HMI Controller.

What’s in this Chapter?

This chapter contains the following topics:

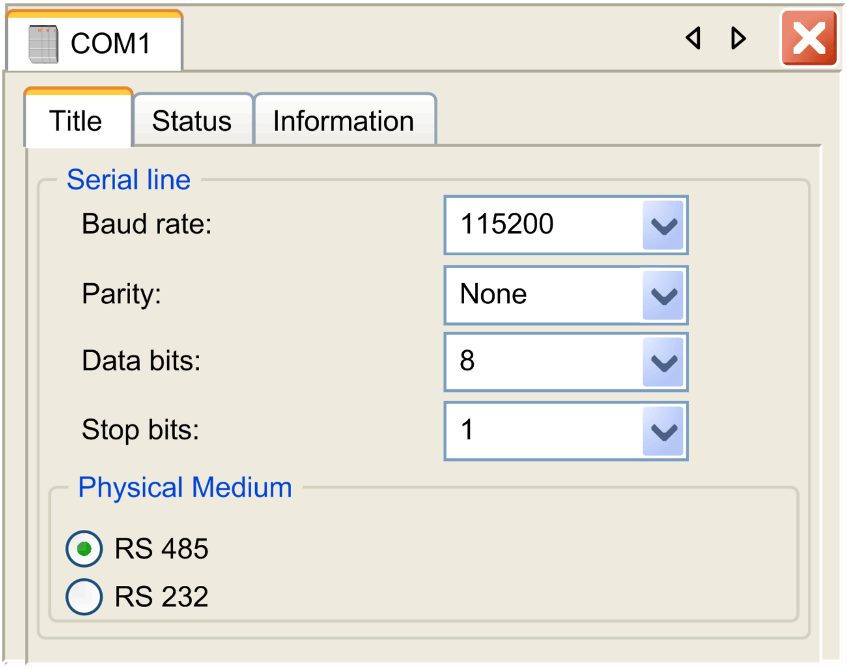
Serial Line Configuration

Introduction

The serial line configuration window allows configuration of the serial line parameters (baud rate, parity, etc.). You can configure up to two serial ports with the XBT GT/GK HMI Controller.

Serial Line Configuration Window

Double click **COM1** or **COM2** in the device tree to access the serial line configuration window:



Serial line

Baud rate: 115200

Parity: None

Data bits: 8

Stop bits: 1

Physical Medium

● RS 485

○ RS 232

The following table provides the description of each parameter:

Parameter	Initial Values	Range	Description
Baud rate	115.2 Kbauds	1.2...115.2 Kbauds	Transmission speed
Parity	None	● None ● Odd ● Even	Used for invalid events detection
Data bits	8	● 7 ● 8	Number of bits for transmitting data
Stop bits	1	● 1 ● 2	Number of stop bits
Physical Medium	RS 485	● RS485 ● RS232	Specify the medium to use

Network Manager

The SoMachine-Network_Manager is automatically added to your project configuration. You can configure two types of **Network_Manager** with the serial line:

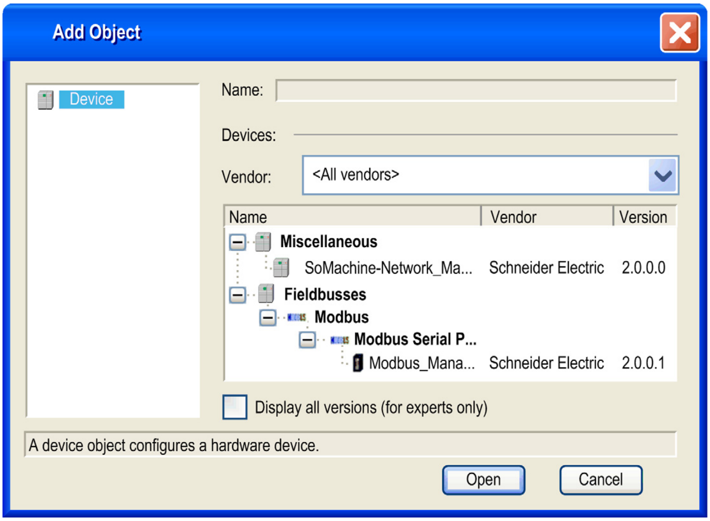
Element	Description
SoMachine-Network_Manager	Used when a XBT GT/GK HMI Controller device is used, or when the Serial Line is also used for PLC programming [1].
Modbus_Manager	Used for Modbus RTU or ASCII protocol in master or slave mode [2].
Legend	
1	Refer to SoMachine *Network_Manager (see page 84)* for additional information on the configuration.
2	Refer to *Modbus Manager (see page 86)* for additional information on the configuration.

NOTE: When using the SoMachine-Network_Manager you can dowload your application to any devices connected to it.

SoMachine Network Manager

Adding a SoMachine Network Manager

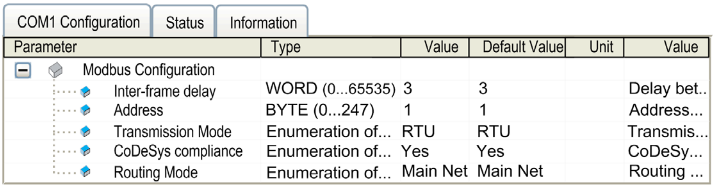
To add a SoMachine Network Manager proceed as explained in the following procedure:

Step	Action
1	Right click **COM1** from the **Devices** window
2	Choose **Add Object...**, **Result:**The following figure appears:  Add Object Device Name: Devices: Vendor: <All vendors> Name Vendor Version Miscellaneous SoMachine-Network_Ma... Schneider Electric 2.0.0.0 Fieldbusses Modbus Modbus Serial P... Modbus_Mana... Schneider Electric 2.0.0.1 □ Display all versions (for experts only) A device object configures a hardware device. Open Cancel
3	Choose the SoMachine Network Manager object and click **Open** **Result:** The SoMachine Network Manager is added in the project structure from the **Devices** window.

NOTE: The Serial Line link does not support both Modbus and SoMachine protocols at the same time.

SoMachine Network Manager Configuration Window

Double click SoMachine Network Manager in the device tree to access the **COM1 Configuration** tab:



Parameter	Type	Value	Default Value	Unit	Value
Modbus Configuration					
Inter-frame delay	WORD (0...65535)	3	3		Delay bet..
Address	BYTE (0...247)	1	1		Address...
Transmission Mode	Enumeration of...	RTU	RTU		Transmis...
CoDeSys compliance	Enumeration of...	Yes	Yes		CoDeSy...
Routing Mode	Enumeration of...	Main Net	Main Net		Routing ...

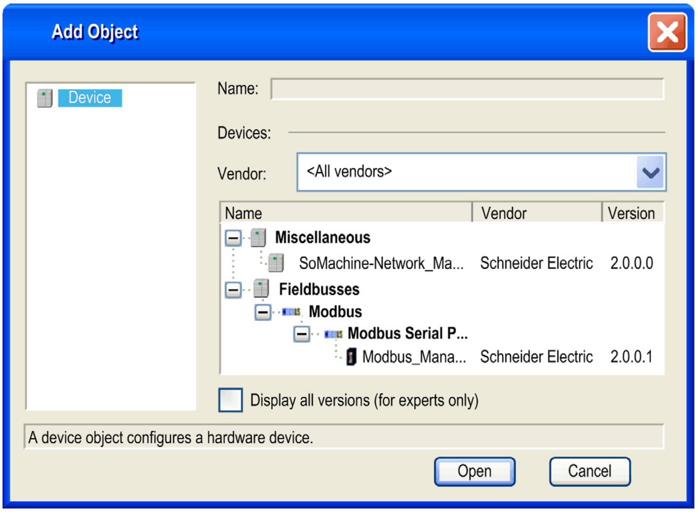
The following table provides the SoMachine-Network_Manager parameters description:

Parameter	Initial Value	Range	Description
Inter-frame delay	3	0...65,535	Delay between frames
Address	1	0...247	Value of the address
Transmission Mode	RTU	RTU ASCII	Transmission mode selection
CoDeSys compliance	Yes	No Yes	CoDeSys compliance
Routing Mode	Main Net	Main Net Sub Net	Routing mode selection

Modbus Manager

Adding a Modbus Network Manager

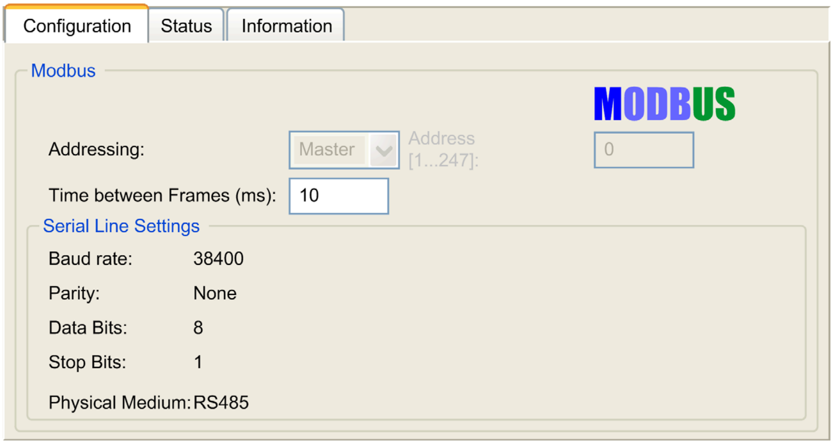
To add a **Modbus-Network_Manager** proceed as explained in the following procedure:

Step	Action
1	Right click **COM1** from the **Devices** window
2	Choose **Add Object...**, **Result:**The following figure appears:  Add Object Device Name: Devices: Vendor: <All vendors> Name Vendor Version Miscellaneous SoMachine-Network_Ma... Schneider Electric 2.0.0.0 Fieldbusses Modbus Modbus Serial P... Modbus_Mana... Schneider Electric 2.0.0.1 □ Display all versions (for experts only) A device object configures a hardware device. Open Cancel
3	Choose the **Modbus-Network_Manager** object and click **Open** **Result:** The **Modbus-Network_Manager** is added in the project structure from the **Devices** window.

NOTE: The Serial Line link does not support both Modbus and SoMachine protocols at the same time.

Modbus Manager Configuration Window

Double click **Modbus_Manager** in the device tree to access the Modbus manager **Configuration** tab:



Modbus

Addressing: Master

Address [1...247]: 0

Time between Frames (ms): 10

Serial Line Settings

Baud rate: 38400

Parity: None

Data Bits: 8

Stop Bits: 1

Physical Medium: RS485

The following table provides the Modbus parameters description:

Element	Description
Modbus Parameters:	
Addressing	Specify the device type: ● Master ● Slave
Address [1...247]	Modbus address of the device.
Time between Frames (ms)	Time required to avoid bus-collision. This parameter must be identical for each Modbus device on the link.
Serial Line Settings:	
Baud Rate	Transmission speed
Parity	Used for error detection
Data Bits	Number of bits for transmitting data
Stop Bits	Number of stop bits
Physical Medium	Medium currently used, it can either be: ● RS485, or ● RS232

Managing Online Applications

11

Connecting the Controller to a PC

Application Transfer

To transfer and run applications, connect your XBT GT/GK HMI Controller to a PC that has SoMachine installed. To transfer an application with an XBT GT/GK HMI Controller, use Ethernet, serial link, USB cables, USB memory key or CF card.

NOTICE

Possible electrical damage to controller components.

Connect the communication cable to the PC before connecting it to the controller.

Failure to follow these instructions can result in equipment damage.

NOTE: Only one XBT GT/GK HMI Controller can be connected to a computer at a time, except when using Ethernet.

Firmware Update

When transferring an application (via Ethernet, USB cables, USB memory key or CF card), the firmware update is done automatically.

USB Cables Requirements

To connect the controller to your PC, specific USB cables are required as shown in the following table:

Product Name	Reference	Description
USB Transfer Cable	XBT ZG 935	Download project data created with the window Editor via the USB interface from the XBT GT/GK HMI Controller unit.
USB Front Cable	XBT ZGUSB	Extension cable attaching USB port to front panel.
USB Front Cable	XBT ZGUSBB	Extension cable attaching USB port to front panel.
USB Programming Cable	TCS XCNA MUM3P	Extension cable attaching USB port to front panel.

NOTE:

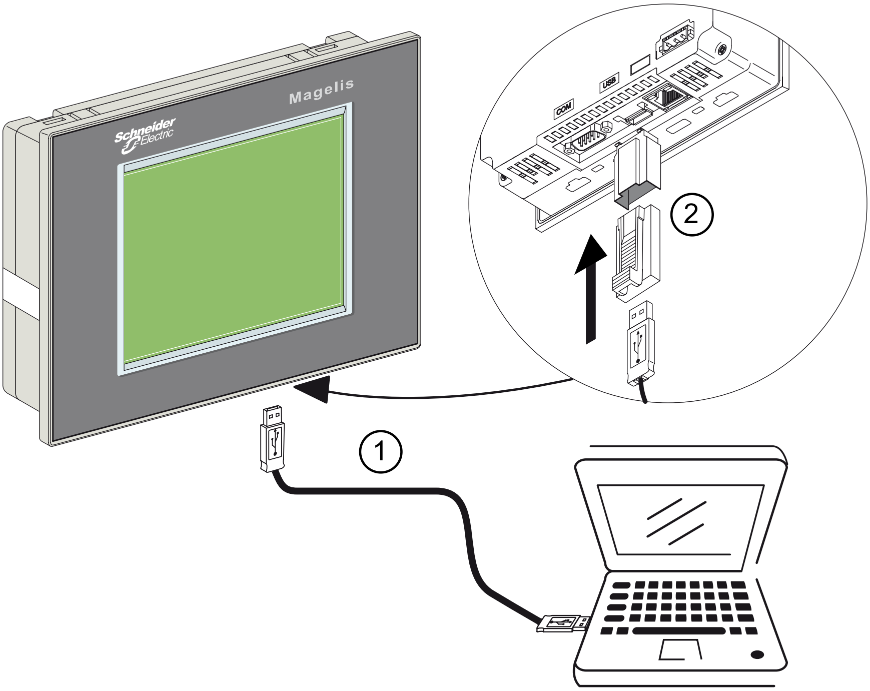
When mounted on a front panel, use the following cables combinations:
- XBT ZG 935 and XBT ZGUSB
- TCS XCNA MUM3P and XBT ZGUSBB

Connecting with USB Cable

To connect the USB cable to your XBT GT/GK HMI Controller, follow the steps in the table below:

Step	Action
1	Connect the USB cable to the XBT GT/GK HMI Controller; check that the USB holder *(see Magelis XBTGC HMI Controller, Hardware Guide)*is in the correct position.
2	Connect your USB cable using the front panel connections *(see Magelis XBTGC HMI Controller, Programming Guide)*.
3	Connect the USB cable to the PC.

The following diagram shows how to connect the XBT GT/GK HMI Controller directly to a PC:

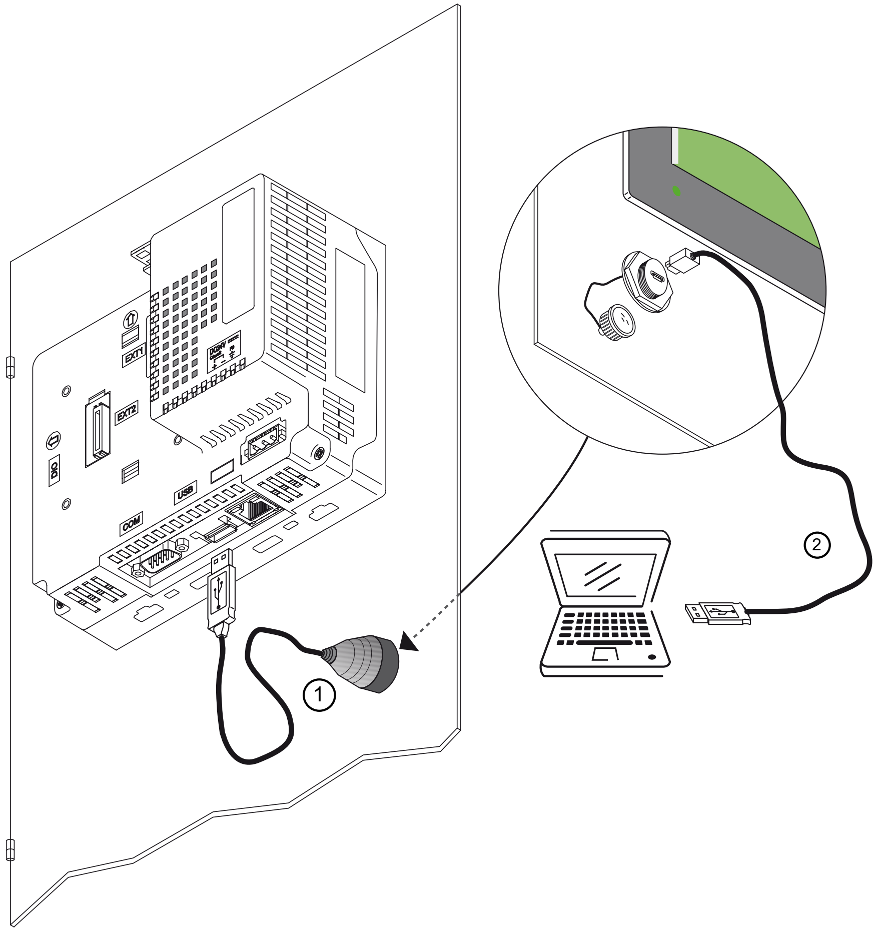


Legend:

1: USB data transfer cable (XBT ZG 935)

2: USB connection on the XBT GT/GK HMI Controller; refer to the XBT GT/GK HMI Controller User Manual *(see Magelis XBTGC HMI Controller, Hardware Guide)* for more information on the USB holder

The following diagram shows how to connect the XBT GT/GK HMI Controller to a PC, when mounted on a front panel:



Legend:

1: USB data transfer cable (XBT ZGUSBB).

2: USB Min B to USB data transfer cable (TCS XCNA MUM3P or XBT ZG 935).

NOTE: An alternative download method consist of connecting your PC to any controllers via USB cable, then connect your XBT GT/GK HMI Controller to the first one via serial link. However transfer speed is slow.

Application Download with Firmware Change

The XBT GT/GK HMI Controller can download an application and change (either upgrade or downgrade) the firmware from a USB memory key. You must first save the application and the appropriate firmware version on a USB memory key.

NOTICE

Loss of Data

Always save your application and firmware version on a USB memory key.

Failure to follow these instructions can result in equipment damage.

To download an application and change the firmware of your controller follow the steps in the table below:

Step	Action
1	Turn off the power supply of your controller, prior connecting the USB memory key.
2	Connect the USB memory key containing the application and firmware into the USB port of your controller.
3	Turn on your controller. **Result:** The application and the firmware version from the USB memory key are downloaded.

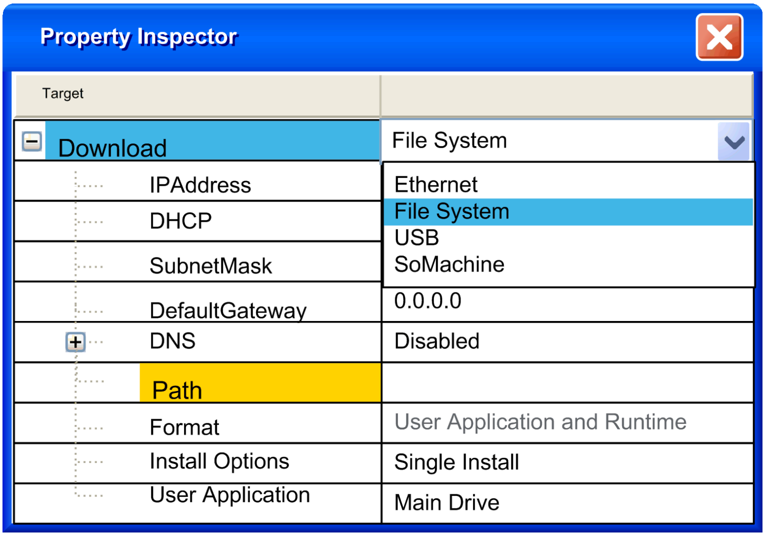
NOTE: If you plug a USB memory key containing the application and firmware while the controller is on, a screen is displayed asking you whether you want to install the application from the USB memory key.

Saving your Application and Firmware on a USB memory key

You can save your application and firmware on a FAT 32 USB memory key. To save, follow the steps in the table below:

Step	Action
1	Insert a USB memory key into the USB port of your computer.
2	Double click on **HMI Application** in the **Devices** window of your project. **Result:** The project switches for the HMI and the main Vijeo Designer window appears.
3	Right click on the controller node in the **Navigator** window, and select **Properties**. **Result:** The **Property Inspector** window appears.

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>4</td><td>Select <b>File System</b> from the <b>Download</b> menu as shown in the following figure:



<table>
<tr><td>Target</td><td></td></tr>
<tr><td>Download</td><td>File System</td></tr>
<tr><td>IPAddress</td><td></td></tr>
<tr><td>DHCP</td><td></td></tr>
<tr><td>SubnetMask</td><td></td></tr>
<tr><td>DefaultGateway</td><td>0.0.0.0</td></tr>
<tr><td>DNS</td><td>Disabled</td></tr>
<tr><td>Path</td><td></td></tr>
<tr><td>Format</td><td>User Application and Runtime</td></tr>
<tr><td>Install Options</td><td>Single Install</td></tr>
<tr><td>User Application</td><td>Main Drive</td></tr>
</table>
</td></tr>
<tr><td>5</td><td>Set the directory from the <b>Path</b> menu to the USB memory key.
<b>NOTE:</b> Select the root level of your USB memory key.</td></tr>
<tr><td>6</td><td>Click the <b>OK</b> button.
<b>Result:</b> The directory is now set to the USB memory key.</td></tr>
<tr><td>7</td><td>Click <b>Build</b> → <b>Download All</b> from the Vijeo Designer main menu bar.
<b>Result:</b> The application is saved onto the USB memory key.</td></tr>
</table>

NOTE: Use a FAT32 USB memory key to save your application and firmware.

Troubleshooting and FAQ

12

Introduction

This chapter contains common troubleshooting procedures and frequently asked questions for the XBT GT/GK HMI Controller.

What's in this Chapter?

This chapter contains the following topics:

Troubleshooting

Introduction

This section lists the possible troubleshooting solutions with the XBT GT/GK HMI Controller, and procedures for troubleshooting them.

Transferring the Application is not Possible

Possible causes:

- PC cannot communicate with the controller.
- SoMachine not configured for the current connection.
- Is your application valid?
- Is the CoDeSys gateway running?
- Is the CoDeSys SP win running?

Resolution:

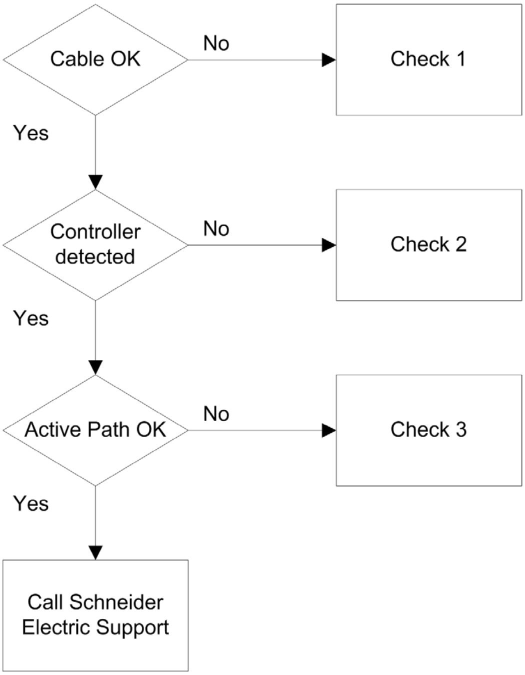
- Refer to Communication between SoMachine and the XBT GT/GK HMI Controller *(see page 96)*.
- Your application program must be valid. Refer to the debugging section in the CoDeSys Online-Help.
- The CoDeSys gateway must be running:
  a. click the CoDeSys Gateway icon in the task bar,
  b. select **Start Gateway**.

Communication Between SoMachine and the XBT GT/GK HMI Controller is not Possible.

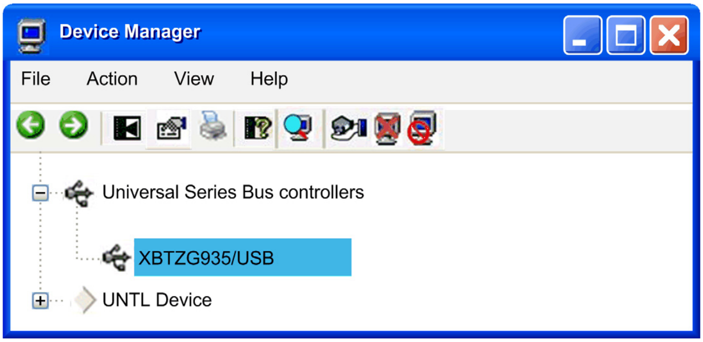
Possible causes:

- SoMachine not configured for the current connection.
- Incorrect cable usage.
- Controller not detected by the PC.
- Communication settings are not correct.
- The controller has detected an error or its firmware is invalid.

Resolution: Follow the flowchart below for troubleshooting purposes and then refer to the next table:



Check	Action
1	Check that: ● The cable is correctly linked to the controller and to the PC, and is not damaged, ● You used the specific cable or adapter, depending on the connection type: ● Ethernet and Serial link connection. ● XBT ZG 935 cable for a USB connection. ● XBT ZG 935 and XBT ZGUSB or TCS XCNA MUM3P and XBT ZGUSBB connection when the controller is mounted on a front panel.

Check	Action
2	Check that the XBT GT/GK HMI Controller has been detected by your PC: **1.** Click **Start** → **Control Panel** → **System**, then select the **Hardware** tab and click **Device Manager**, **2.** Check that the XBT GT/GK HMI Controller node appears in the list, as shown below:  Device Manager File Action View Help Universal Series Bus controllers XBTZG935/USB UNTL Device **3.** If the XBT GT/GK HMI Controller node does not appear, or if there is an ? icon in front of the node, disconnect and reconnect the cable on the controller side.
3	Check that the active path is correct: **1.** Double click the controller node in the device view. **2.** Check that the XBT GT/GK HMI Controller node appears in bold, not in italic. If not: **a.** Stop the CoDeSys Gateway: right click the icon in the task bar and select **Stop Gateway**. **b.** Disconnect and reconnect the cable on the controller side. **c.** Start the CoDeSys Gateway: right click the icon in the task bar and select **Start Gateway**. **d.** Select the gateway in the controller window of SoMachine and click **Scan network**. Select the XBT GT/GK HMI Controller node and click **Set active path**. **NOTE:** If your PC is connected to an Ethernet network, its address might have changed. In this case, the currently set active path is no longer correct and the XBT GT/GK HMI Controller node appears in italics. Select the XBT GT/GK HMI Controller node and click **Resolve Name**. When the node no longer appears in italics. To correct this, click **Set Active Path**.

Application Does Not Go To RUN State

Possible causes:

No POU declared in the task.

ControllerLockout activated.

Resolution:

As POUs are managed by tasks, add a POU to a task:

1. Double click a task in the device view.
2. Click **Add POU** in the task window.
3. Select the POU you want to execute in the **Input Assistant** window and click **OK**.
4. Unlock ControllerLockout in Vijeo Designer.

Creating the Boot Application is not Possible

Possible cause:

Operation not possible while the controller is in RUN state.

Resolution:

- Select **Stop Application**.
- Select **Create Boot Project**.

Changing Device Name does not work

Possible cause:

Application is running.

Resolution:

- Select **Stop Application**,
- Change device name.

CANopen Heartbeat is not sent on a regular basis

Possible cause:

Heartbeat value is not correct.

Resolution:

The Heartbeat of the CANopen master must be reset:

- Calculate the Heartbeat consumer time:
  Heartbeat Consumer Time = Producer Time * 1.5
- Update the Heartbeat value

Monitoring of the POU is slow

Possible cause:

- Task interval is too small or POU is too big.
- Connection speed low between controller and device (over serial connection).

Resolution:

- Increase the configured task interval.
- Split the application in the smaller POUs.

Out of Memory appears on the HMI screen

Possible cause:

- The number of variables and symbols shared between the controller and the HMI is too high.

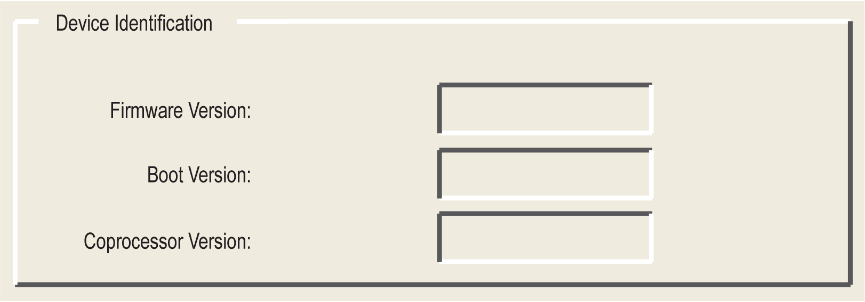
Resolution:

- Decrease the number of variables and symbols shared between the controller and the HMI.
- Power cycle the HMI.

Frequently Asked Questions

How can I Determine the Firmware, Boot and Chip Version of the Controller?

In online mode, double click the controller node in the device view. In the controller window, select the **Service** tab. The device identification area provides version information:



Device Identification

Firmware Version:

Boot Version:

Coprocessor Version:

What Programming Languages are supported by a XBT GT/GK HMI Controller?

The following languages are supported:

- Continuous Function Chart (CFC)
- Function Block Diagram (FBD)
- Instruction List (IL)
- Ladder Logic Diagram (LLD)
- Sequential Function Chart (SFC)
- Structured Text (ST)

What Variable Types are supported by an XBT GT/GK HMI Controller Controller?

Refer to the Supported Variables section *(see page 26)*.

Can I Use SoMachine Network to Communicate with an Equipment Connected to the Serial Line of my XBT GT/GK HMI Controller?

It is possible with an XBT GC 2230 HMI Controller only if the serial line is configured with the SoMachine Network Protocol *(see page 81)*.

Limitations:

- Slow access to the remote equipment.
- You cannot cascade to other equipment.

For more information, refer to SoMachine - Network/Combo: Vijeo-Designer part, available in the appendix of XBT GC online help.

When should I use Freewheeling or Cyclic Mode?

- Freewheeling: use this mode if you accept a variable cycle time. The next cycle starts after a waiting duration that equals 30% of the last cycle execution time.
- Cyclic: use this mode if you want to control the frequency cycle.

What does the Start all applications after download or online change checkbox do?

- Case 1: Standalone HMI application download or HMI and Control applications download
  - The BOOT state of the Control application is updated based on the checkbox setting
- Case 2: Control application download only
  - The setting of the checkbox takes effect after the download/online change
  - The RUN of the control application at BOOT time is not affected

Can I connect the PC (SoMachine) and the controller through Ethernet and TwidoPort?

No, because Twidoport only supports the Modbus protocol.

Can I connect several XBT GT/GK HMI Controller through several USB ports of my PC?

No, this is not supported.

Why does Source Download lead to Communication Interruptions?

Because this function is not supported by XBT GT/GK HMI Controller. To connect to an XBT GT/GK HMI Controller, you must have the source of the controller application on the PC with SoMachine.

Why does the communication between the HMI and SoMachine get interrupted?

Making online changes to an XBT GT application interrupts the communication between the HMI and SoMachine and the following message appears: "Online change in execution". The interruption is proportional to the number of online changes you made.

When I use a new Modicon M238 Logic Controller with a previous HMI application, I cannot communicate anymore?

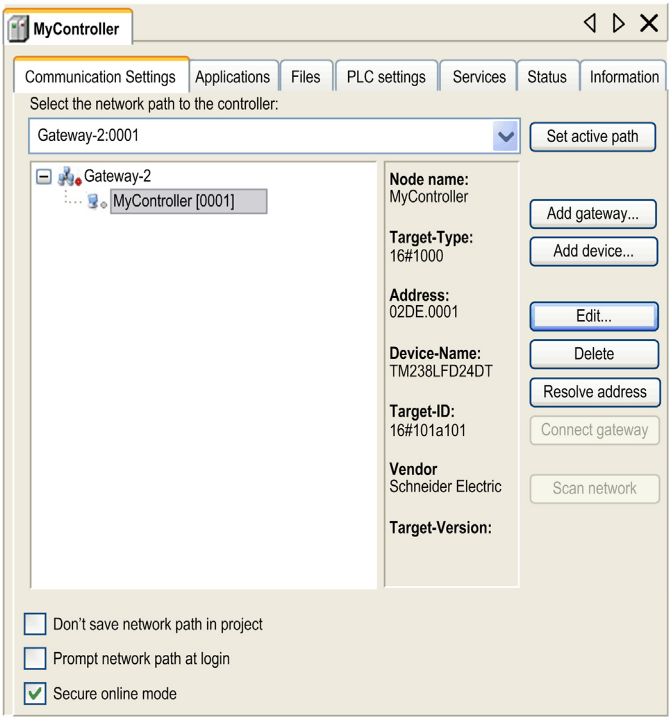
This is because the new controller name in the HMI application (Vijeo-Designer) is not updated. The HMI application is configured with the previous controller name.

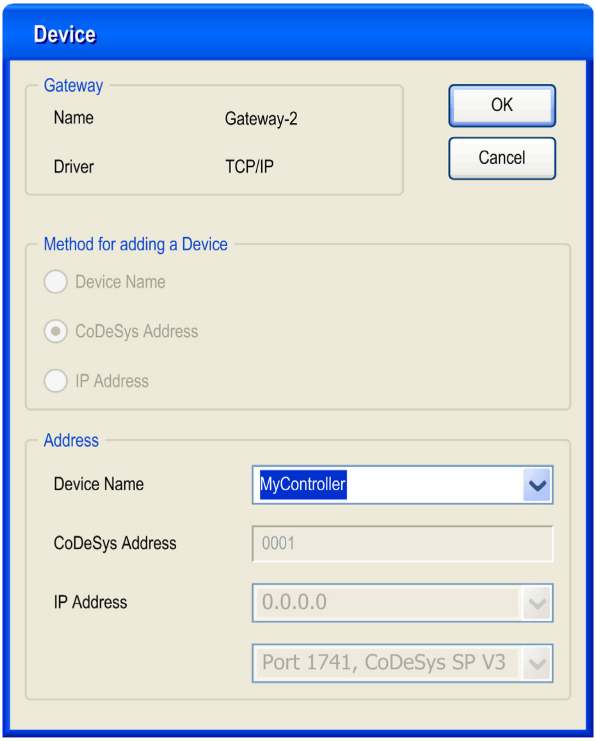
To update the controller name you can:

- Update manually *(see page 102)* the controller name in the HMI application to be consistent with the controller name used in SoMachine, or,
- Update manually *(see page 105)* the controller name in SoMachine to be consistent with the controller name used in the HMI application (Vijeo-Designer), or,
- Create a generic application *(see page 107)* that can be easily used without modifying the HMI application.

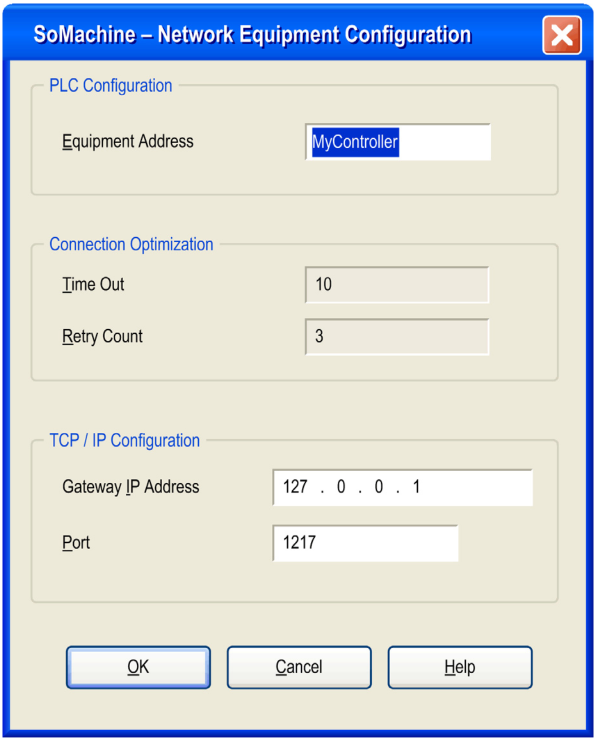
How do I manually update the Modicon M238 Logic Controller name in my HMI application using Vijeo-Designer?

Copy the controller device name from SoMachine as explained in the procedure below:

Step	Action
1	Double click on the controller node from the SoMachine **Devices** window **Result:** The **Device Editor** window opens
2	Select the **Communication Settings** tab
3	Select your controller available in the gateway, as shown in the next figure  MyController Communication Settings Applications Files PLC settings Services Status Information Set active path Add gateway... Add device... Edit... Delete Resolve address Connect gateway Scan network Select the network path to the controller: Gateway-2:0001 Gateway-2 MyController [0001] **Node name:** MyController **Target-Type:** 16#1000 **Address:** 02DE.0001 **Device-Name:** TM238LFD24DT **Target-ID:** 16#101a101 **Vendor** Schneider Electric **Target-Version:** □ Don't save network path in project □ Prompt network path at login ☑ Secure online mode

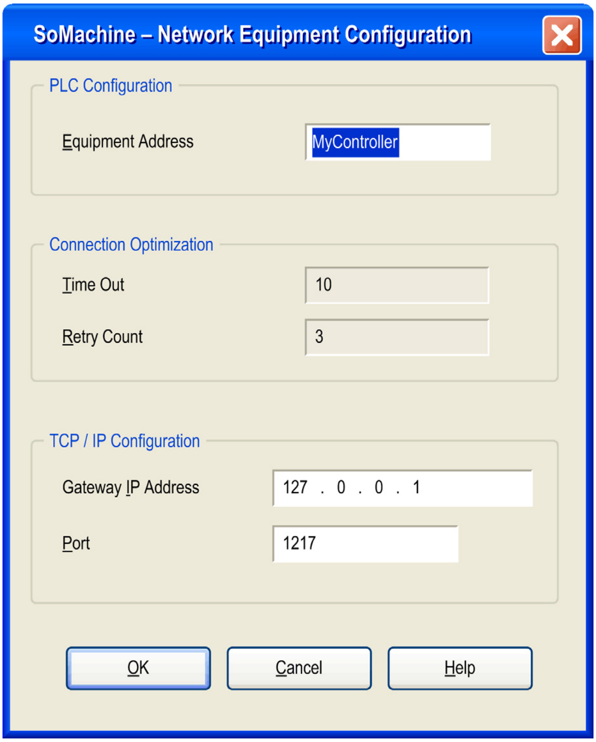
Step	Action
4	Click **Edit...**, the **Device** window opens:  Device Gateway Name: Gateway-2 Driver: TCP/IP Method for adding a Device ○ Device Name ● CoDeSys Address ○ IP Address Address Device Name: MyController CoDeSys Address: 0001 IP Address: 0.0.0.0 Port 1741, CoDeSys SP V3 OK Cancel
5	Copy the controller name available in the **Address** → **Device Name** field

Paste the controller device name into Vijeo-Designer **Network Equipment Configuration** window, as explained in the procedure below:

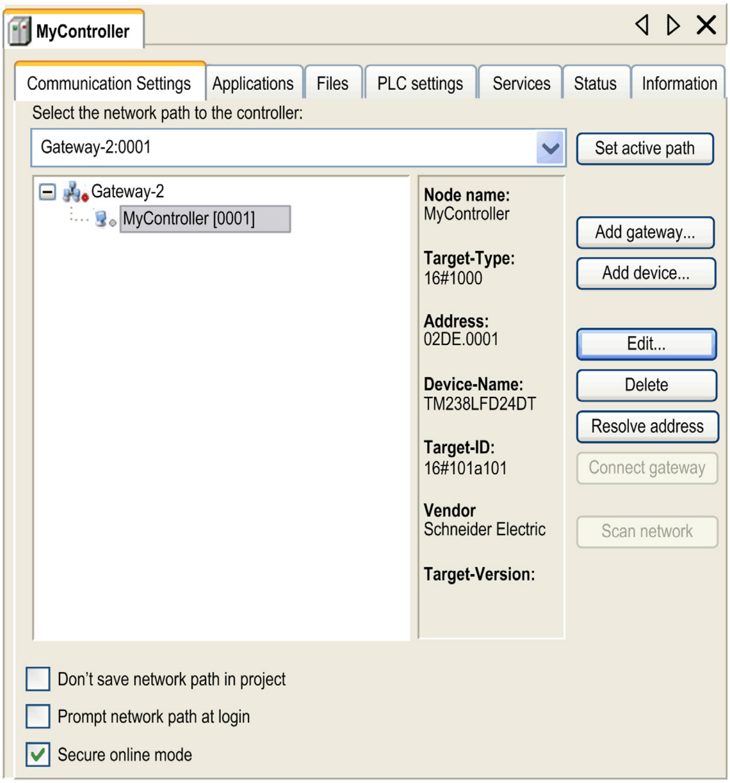
Step	Action
1	Double click **IO Manager** → **SoMachineNetwork** → **Your controller** from the Vijeo-Designer **Navigator** **Result:** The **Network Equipment Configuration** window opens
2	Paste the controller device name in **PLC Configuration** → **Equipment Address** field  SoMachine – Network Equipment Configuration PLC Configuration Equipment Address: MyController Connection Optimization Time Out: 10 Retry Count: 3 TCP / IP Configuration Gateway IP Address: 127 . 0 . 0 . 1 Port: 1217
3	Click the **OK** button
4	Double click **IO Manager** → **SoMachineNetwork** → **Your HMI** from the Vijeo-Designer **Navigator** **Result:** The **Network Equipment Configuration** window opens
5	Paste the controller device name in **PLC Configuration** → **Equipment Address** field
6	Click the **OK** button
7	Download the application into your HMI device **Result:** Your HMI device name is now updated

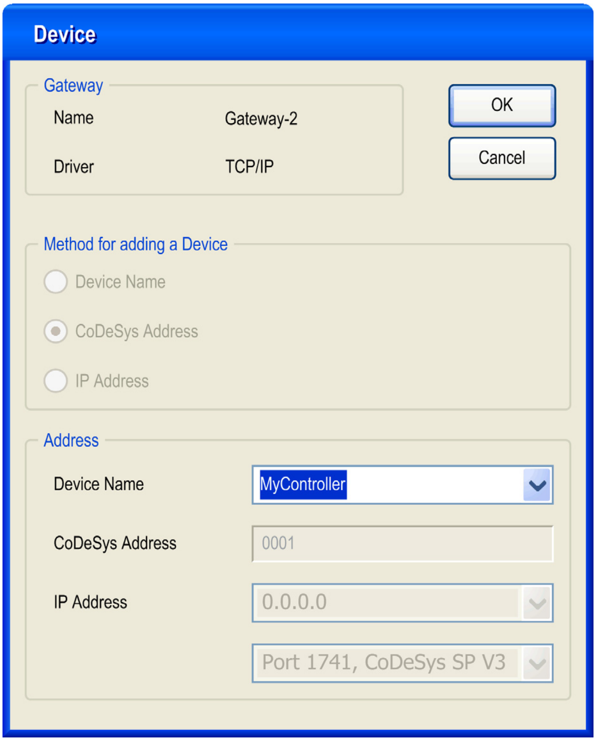
How do I manually update the Modicon M238 Logic Controller name in my SoMachine application using Vijeo-Designer?

Paste the controller device name from Vijeo-Designer as explained in the procedure below:

Step	Action
1	Double click **IO Manager** → **SoMachineNetwork** → **Your controller** from the Vijeo-Designer **Navigator** **Result:** The **Network Equipment Configuration** window opens
2	Copy the controller device name in **PLC Configuration** → **Equipment Address** field  SoMachine – Network Equipment Configuration PLC Configuration Equipment Address: MyController Connection Optimization Time Out: 10 Retry Count: 3 TCP / IP Configuration Gateway IP Address: 127 . 0 . 0 . 1 Port: 1217
3	Click the **OK** button

Copy the controller device name into the SoMachine **Device Editor** as explained in the procedure below:

Step	Action
1	Double click on the controller node from the SoMachine **Devices** window **Result:** The **Device Editor** window opens
2	Select the **Communication Settings** tab
3	Select your controller available in the gateway, as shown in the next figure  MyController Select the network path to the controller: Gateway-2:0001 Gateway-2 MyController [0001] **Node name:** MyController **Target-Type:** 16#1000 **Address:** 02DE.0001 **Device-Name:** TM238LFD24DT **Target-ID:** 16#101a101 **Vendor** Schneider Electric **Target-Version:** □ Don't save network path in project □ Prompt network path at login ☑ Secure online mode Communication Settings Applications Files PLC settings Services Status Information Set active path Add gateway... Add device... Edit... Delete Resolve address Connect gateway Scan network

Step	Action
4	Click **Edit...**, the **Device** window opens:  Device Gateway Name Gateway-2 Driver TCP/IP Method for adding a Device ○ Device Name ● CoDeSys Address ○ IP Address Address Device Name MyController CoDeSys Address 0001 IP Address 0.0.0.0 Port 1741, CoDeSys SP V3
5	Paste the controller name available in the **Address** → **Device Name** field
6	Click the **OK** button **Result:** Your controller device name is now updated

How do I create a generic application ?

The following procedure shows the main steps for creating and using generic application. Note that all these steps are explained in the next paragraphs:

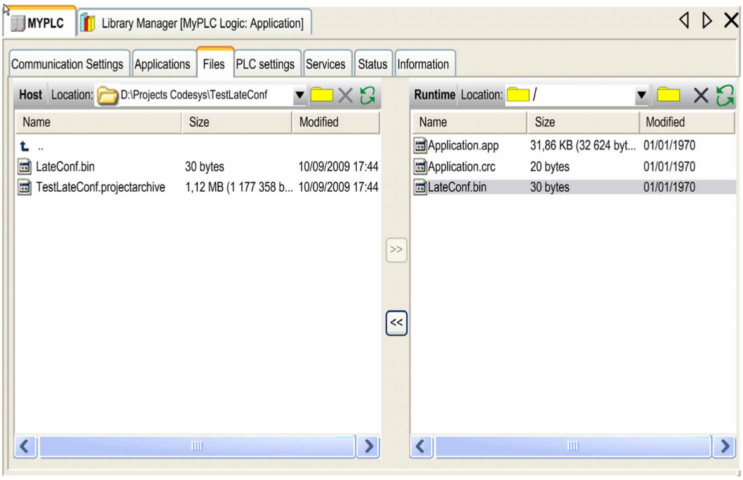
Step	Action
1	Create a project *(see page 108)* archive file
2	Save the controller device name *(see page 108)* used by the HMI device
3	Extract the project archive file
4	Copy the controller device name *(see page 109)* saved in step 2
5	Use the application with the new controller and the existing HMI application

Create a Project Archive File

Create a project archive file by selecting **File** → **Project Archive** → **Save/Send Archive** from the SoMachine menu.

Save the Controller Device Name

Follow the procedure below to save the controller device name (**LateConf.bin**) locally:

Step	Action
1	Double click on the controller node from the SoMachine **Devices** window **Result:** The **Device Editor** window opens
2	Check that the controller is set as the **Active Path** from the **Communication Settings** tab
3	Refresh the **Runtime** browser located in the **Files** tab by clicking the **Refresh** button **Result:** The **Runtime** browser is refreshed and a **LateConf.bin** file appears
4	Select the **LateConf.bin** file from the **Runtime** browser
5	Select a location on your local computer from the **Host** browser
6	Click the **Left Arrow** button (button highlighted in the picture) to move the **LateConf.bin** file from the **Runtime** browser into the **Host** browser  MYPLC Library Manager [MyPLC Logic: Application] Communication Settings Applications Files PLC settings Services Status Information >> << Host Location: D:\Projects Codesys\TestLateConf Name Size Modified .. LateConf.bin 30 bytes 10/09/2009 17:44 TestLateConf.projectarchive 1,12 MB (1 177 358 b... 10/09/2009 17:44 Runtime Location: / Name Size Modified Application.app 31,86 KB (32 624 byt... 01/01/1970 Application.crc 20 bytes 01/01/1970 LateConf.bin 30 bytes 01/01/1970 **Result:** The **LateConf.bin** file is saved onto your local computer

At this stage, the project archive file and the **LateConf.bin** file are available locally on your computer. They can be used as a generic application on other controllers and computer after their extraction.

Copy the Controller Device Name

Follow the procedure below to copy the controller device name (**LateConf.bin**):

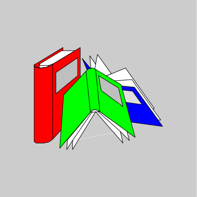
Step	Action
1	Double click on the controller node from the SoMachine window **Result:** The **Device Editor** window opens
2	Check that the controller is set as the **Active Path** from the **Communication Settings** tab
3	Locate the **LateConf.bin** file in the **Host** browser from the **File**tab
4	Select the **LateConf.bin** file from the **Host** browser
5	Click the **Right Arrow** button to move the **LateConf.bin** file from the **Host** browser into the **Runtime** browser **Result:** The **LateConf.bin** file is saved onto your new controller
6	Power cycle your controller to updated its name

Why does the Task Monitor always show zero ms for the Average and Minimum Task Times?

The XBTGC only supports reporting back of cycle times to a 1 ms resolution, and requires a minimum of 2 ms for one HMI with a Control Process cycle (since the CPU is scheduled to give HMI and Control each 1 ms per 2 ms).

If a task requires less than 2 ms (2000 µs) to run, the Task Monitor will show 0 µs.

Glossary



0-9

%I

According to the IEC standard, %I represents an input bit (for example a language object of type digital IN).

%IW

According to the IEC standard, %IW represents an input word register (for example a language object of type analog IN).

%Q

According to the IEC standard, %Q represents an output bit (for example a language object of type digital OUT).

%QW

According to the IEC standard, %QW represents an output word register (for example a language objet of type analog OUT).

1-phase counter

A *1-phase counter* uses 1 hardware input as counter input. It usually counts up or counts down when there is pulse signal in the input.

2-phase counter

A *2-phase counter* uses the phase difference between 2 input counter signals to count up or count down.

A

ASCII

The *american standard code for information interchange* is a communication protocol for representing alphanumeric characters (letters, numbers, and certain graphic and control characters).

B

BCD

The *binary coded decimal format* represents decimal numbers between 0 and 9 with a set of 4 bits (a nybble/nibble, also titled as Halfbyte). In this format, the 4 bits used to encode decimal numbers have an unused range of combinations. For example, the number 2,450 is encoded as `0010 0100 0101 0000`

BOOL

A *Boolean* type is the basic data type in computing. A `BOOL` variable can have one of these values: 0 (`FALSE`), 1 (`TRUE`). A bit that is extracted from a word is of type `BOOL`, for example: `%MW10.4` is a fifth bit a memory word number 10.

Boot application

Files that contain machine dependent parameters:

- machine name
- device name or IP address
- Modbus Serial Line address
- Routing table

BOOTP

The *bootstrap protocol* is a UDP network protocol that can be used by a network client to automatically obtain an IP address (and possibly other data) from a server. The client identifies itself to the server using the client's MAC address. The server—which maintains a pre-configured table of client device MAC addresses and associated IP addresses—sends the client its pre-configured IP address. BOOTP was originally used as a method that enabled diskless hosts to be remotely booted over a network. The BOOTP process assigns an infinite lease of an IP address. The BOOTP service utilizes UDP ports 67 and 68.

BYTE

When 8 bits are grouped together, they are called a `BYTE`. You can enter a `BYTE` either in binary mode or in base 8. The `BYTE` type is encoded in an 8-bit format that ranges from `16#00` to `16#FF` (in hexadecimal format).

C

CANopen

CANopen is an open industry-standard communication protocol and device profile specification.

CFC

The *continuous function chart* (an extension of the IEC61131-3 standard) is a graphical programming language that works like a flowchart. By adding simple logicals blocks (AND, OR, etc.), each function or function block in the program is represented in this graphical format. For each block, the inputs are on the left and the outputs on the right. Block outputs can be linked to inputs of other blocks in order to create complex expressions.

controller

A *controller* (or “programmable logic controller,” or “programmable controller”) is used to automate industrial processes.

CRC

A network message’s *cyclic redundancy check* field contains a small number of bits that produce a checksum. The message is calculated by the transmitter according to the message’s content. Receiving nodes then recalculate the field. Any discrepancy in the two CRC fields indicates that the transmitted message and the received message are different.

D

DHCP

The *dynamic host configuration protocol* is an advanced extension of BOOTP. DHCP is a more advanced, but both DHCP and BOOTP are common. (DHCP can handle BOOTP client requests.)

DINT

A *double integer* type is encoded in a 32-bit format.

DNS

The *domain name system* is the naming system for computers and devices connected to a LAN or the Internet.

DWORD

A *double word* type is encoded in a 32-bit format.

E

expansion bus

The *expansion bus* is an electronic communication bus between expansion modules and a CPU.

expansion I/O module

An *expansion input or output module* is either a digital or analog module that adds additional I/O to the base controller.

F

FB

A *function block* performs a specific automation function, such as speed control, interval control, or counting. A function block comprises configuration data and a set of operating parameters.

FBD

A *function block diagram* is a graphically oriented programming language, compliant with IEC 61131-3. It works with a list of networks whereby each network contains a graphical structure of boxes and connection lines which represents either a logical or arithmetic expression, the call of a function block, a jump, or a return instruction.

firmware

The *firmware* represents the operating system on a controller.

G

GVL

The *global variable list* manages global variables that are available in every application POU.

H

HMI

A *human-machine interface* is an operator interface (usually graphical) for industrial equipment.

HSC

high-speed counter

I

IEC 61131-3

The IEC 61131-3 is an *international electrotechnical commission* standard for industrial automation equipment (like controllers). IEC 61131-3 deals with controller programming languages and defines 2 graphical and 2 textual programming language standards:

- **graphical:** ladder diagram, function block diagram
- **textual:** structured text, instruction list

IEEE

The *institute of electrical and electronics engineers* is a non-profit international standards and conformity assessment body for advances in all fields of electrotechnology.

IEEE 802.3

IEEE 802.3 is a collection of IEEE standards defining the physical layer, and the media access control (MAC) sublayer of the data link layer, of wired Ethernet.

IL

A program written in the *instruction list* language is composed of a series of instructions executed sequentially by the controller. Each instruction includes a line number, an instruction code, and an operand. (IL is IEC 61131-3 compliant.)

INT

A single *integer* is encoded in 16 bits.

L

latching input

A *latching input* module interfaces with devices that transmit messages in short pulses. Incoming pulses are captured and recorded for later examination by the application.

LCD

liquid crystal display

LD

A program in the *ladder diagram* language includes a graphical representation of instructions of a controller program with symbols for contacts, coils, and blocks in a series of rungs executed sequentially by a controller. IEC 61131-3 compliant.

M

MAC address

The *media access control address* is a unique 48-bit number associated with a specific piece of hardware. The MAC address is programmed into each network card or device when it is manufactured.

MAST

A master (MAST) task is a processor task that is run through its programming software. The MAST task has two sections:

- **IN:** Inputs are copied to the IN section before execution of the MAST task.
- **OUT:** Outputs are copied to the OUT section after execution of the MAST task.

master/slave

The single direction of control in a network that implements the master/slave model is always from a master device or process to one or more slave devices.

Modbus

The Modbus communication protocol allows communications between many devices connected to the same network.

N

network

A network includes interconnected devices that share a common data path and protocol for communications.

NMT

Network management protocols provide services for network initialization, error control, and device status control.

node

A *node* is an addressable device on a communication network.

P

PDO

A *process data object* is transmitted as an unconfirmed broadcast message or sent from a producer device to a consumer device in a CAN-based network. The transmit PDO from the producer device has a specific identifier that corresponds to the receive PDO of the consumer devices.

PLI

pulse latch input

POU

A *program organization unit* includes a variable declaration in source code and the corresponding instruction set. POUs facilitate the modular reuse of software programs, functions, and function blocks. Once declared, POUs are available to one another. SoMachine programming requires the utilization of POUs.

protocol

A *protocol* is a convention or standard that controls or enables the connection, communication, and data transfer between two computing endpoints.

PTO

Pulse train outputs are used to control for instance stepper motors in open loop.

PWM

Pulse width modulation is used for regulation processes (e.g. actuators for temperature control) where a pulse signal is modulated in its length. For these kind of signals, transistor outputs are used.

R

RAM

random access memory

REAL

Real is a numeric data type. The REAL type is encoded in a 32-bit format.

RTU

A *remote terminal unit* is a device that interfaces with objects in the physical world to a distributed control system or SCADA system by transmitting telemetry data to the system and/or altering the state of connected objects based on control messages received from the system.

S

SCADA

A *supervisory control and data acquisition* system monitors, manages, and controls industrial applications or processes.

scan

A controller’s scanning program performs 3 basic functions: [1] It reads inputs and places these values in memory; [2] it executes the application program 1 instruction at a time and stores results in memory; [3] It uses the results to update outputs.

SDO

A *service data object* message is used by the fieldbus master to access (read/write) the object directories of network nodes in CAN-based networks. SDO types include service SDOs (SSDOs) and client SDOs (CSDOs).

SFC

A program written in the *sequential function chart* language can be used for processes that can be split into steps. SFC is composed of steps with associated actions, transitions with associated logic condition, and directed links between steps and transitions. (The SFC standard is defined in IEC 848. It is IEC 61131-3 compliant.)

STN

scan twisted nematics (also known as passive matrix)

Structured Text

A program written in the *structured text* (ST) language includes complex statements and nested instructions (such as iteration loops, conditional executions, or functions). ST is compliant with IEC 61131-3.

symbol

A *symbol* is a string of a maximum of 32 alphanumeric characters, of which the first character is alphabetic. It allows you to personalize a controller object to facilitate the maintainability of the application.

T

task

A group of sections and subroutines, executed cyclically or periodically for the MAST task, or periodically for the FAST task.

A task possesses a level of priority and is linked to inputs and outputs of the controller. These I/O are refreshed in consequence.

A controller can have sereval tasks.

U

UDINT

This abbreviation for an *unsigned double integer* (encoded in 32 bits).

UINT

An *unsigned integer* is encoded in 16 bits.

Index